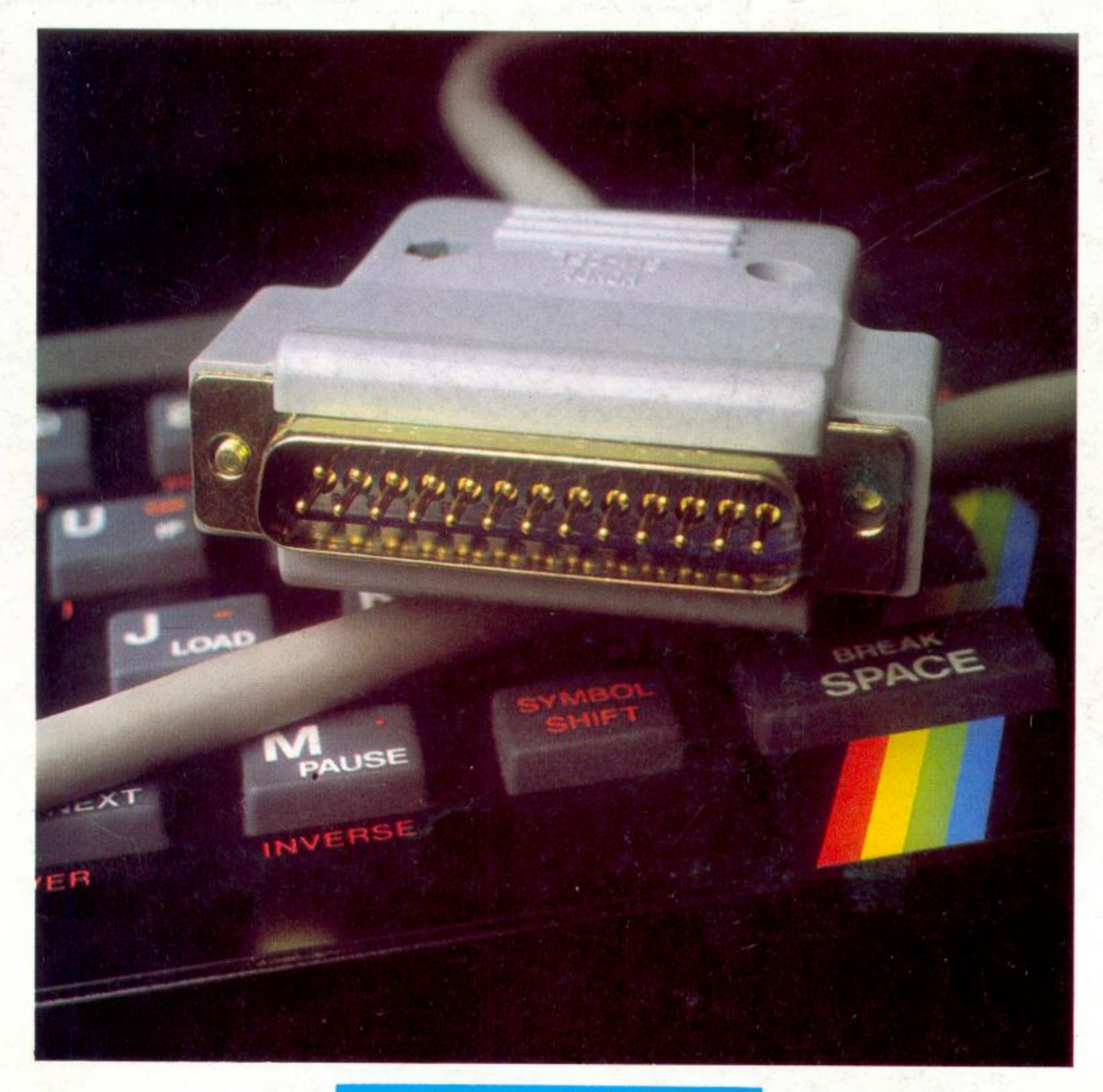
SEPTIEMBRE 86 - 300 ptas.

TodoSpectrum

AÑO II - Número 24

REVISTA EXCLUSIVA PARA USUARIOS



CONSTRUIMOS

INTERFACE SERIE RS-232-C

DESCUBRIMOS

Guia del Hacker: Alien Highway

PROBAMOS

Diseño de Sprites

PROGRAMAMOS

Inversion de UDGs

INVESTIGAMOS

64 Columnas en Pascal

SUMARIO

AÑO II - N.º 24 - SEPTIEMBRE 1986

SERVICIO DE EJEMPLARES ATRASADOS



Para hacer su pedido, rellene este cupón **HOY MISMO** y envíelo a:

Todospectrum Bravo Murillo, 377
Tel. 733 96 62 - 28020 MADRID

Ruego me envíen los siguientes ejemplares atrasados de TODOSPECTRUM .... al precio de 300 pts.

El importe lo abonaré
☐ POR CHEQUE ☐ CONTRA REEMBOLSO ☐ CON MI TARJETA DE CREDITO ☐ AMERICAN EXPRESS ☐ VISA ☐ INTERBANK

Número de mi tarjeta: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Fecha de caducidad .... Firma

NOMBRE ....

DIRECCION ....

CIUDAD .... C. P. ....

PROVINCIA ....

Complete su colección de

Todospectrum

A continuación le resumimos el contenido de los ejemplares aparecidos hasta ahora.

Núm. 2 - 300 ptas.
Gráficos profesionales/Desplazamiento pixel a pixel/Utilización de rutinas/Construcción del interface centronics/Programas de utilidad para microdrive/Rutina reset en código máquina/Análisis del editor de textos Tasword/Interfaces para impresoras/Programas.

Núm. 3 - 300 ptas.
Novedades sonimag'84/Ampliando el Basic/Programas para ordenar programas/Gráficos con el VU-3D/Lenguaje Forth/Archivos en microdrive/Programación de un interface de impresora/Programas.

Núm. 4 - 300 ptas.
De profesión: programador/Consola para el Spectrum/Comparación código máquina-Basic/Análisis programa contabilidad/Calendario/Pascal/Programas.

Núm. 5 - 300 ptas.
Floppys para Spectrum/Diseño asistido por ordenador/64 Caracteres por línea/Juego de la vida/Pascal/Así hacemos las portadas/Control de evaluaciones/Programas.

Núm. 6 - 300 ptas.
Representación de funciones/Todos los caminos conducen a la ROM/Juegos/Pascal/Construcción de un lápiz óptico/Programas de gestión. El SITI/Logo: torgugas para todos/ Interrupciones del Z-80/Programas.

Núm. 7 - 300 ptas.
Del 48 al PLUS paso a paso/¿Plotter para Spectrum?/Juegos/Libros de código máquina/Lápiz óptico. Programación del montaje/El LOGO en la escuela/Pascal/Floppys para Spectrum/Programas.

Núm 8 - 300 ptas.
Amplía tu memoria... a 48 K/Arquitectura: análisis del PREYME/Juegos/FORTH. Nociones básicas/Una clave, please/QL Magazine. Ultimas novedades, análisis de software, Lenguajes/Aula informática con Spectrum/Programas.

Núm 9 - 300 ptas.
Spectrum parlanchín/Juegos/Aula informática con Spectrum/Análisis: Comercial 4/Pascal/Periféricos: Wafadrive/QL Magazine: EASEL lo mejor de PSION. Música con QL/Desplazamiento Pixel a Pixel, aportación de lectores/Programas/Programer II.

Núm. 10 - 300 ptas.
Discos: invesdisc 200/Juegos/Dos programas simultáneos/Protección del software/Conozca extremadura, consulte a su ordenador/Desensamblador Z-80/Sofware educativo/QL Magazine: novedades Informat, Hoja de cálculo, Ajedrez/Construya su propio Joystick/Pascal/programas.

DISPONEMOS DE TAPAS ESPECIALES PARA SUS EJEMPLARES DE ZX
(sin necesidad de encuadernación)

Núm. 11 - 300 ptas.
Actualidad/La otra cara del LOGO/Juegos/El Spectrum habla castellano/SOFTAID ayuda para Etiopía/S.O.S. aquí el Spectrum/Dibujar con lápiz óptico/QL Magazine: Procesador de textos. Teclas de función programables/Programas.

Núm. 12 - 300 ptas.
Actualidad/Inteligencia artificial/Lápiz óptico dk'TRONICS/Juegos/Análisis/Bingo/Z-80 PIO/Código máquina/Análisis: MASTERFILE/Programas.

Núm. 13 - 300 ptas.
Actualidad/Discos: Discovery 1/Juegos/Inteligencia artificial/Un nuevo sistema operativo/QL Magazine: Archive, Cartridge doctor. Aplicaciones comerciales/Código máquina/Programas.

Núm. 14 - 300 ptas.
Actualidad, Spectrum 128/Cálculo de estructuras para ingenieros y arquitectos/HELP utilidades en microdrive/Juegos/El microdrive ese desconocido/Código máquina/QL Magazine: GRAPHIC QL. Juegos. Discos de 720 K/Un nuevo operativo/Programas.

Núm. 15 - 300 ptas.
Actualidad/Spectrum 128/Un nuevo operativo/Circulos redondos/Juegos/Utilidades: BETA-BASIC/QL Magazine: Introducción al SUPER BASIC. Nuevas utilidades/Hardware: Puertas lógicas/Código máquina/Programas.

Núm. 16 - 300 ptas.
Actualidad/Cinco horas con SCREEN$/Hardware práctico/Cálculos de infinita precisión/Juegos/Un nuevo operativo/QL Magazine: Gráficos en SUPER-BASIC. Dibujando con ratón. Archivos con Archive. Programa/La última batalla, Juego estratégico.

Núm. 17 - 300 ptas.
Actualidad/Gráficos interactivos/Juegos/Código máquina/Un nuevo operativo/Trucos de programación/QL Magazine: Radiografía del QL. Gráficos en SUPER-BASIC/Libros/Programmas.

Núm. 18 - 300 ptas.
Actualidad/Introducción al C/Libros/Juegos/De cinta a microcinta/Visión panorámica de los microprocesadores más comunes/QL Magazine: Copy de grises. Microprocesadores 68000, una familia numerosa/Curioseando en la ROM/Programas.

DIRECTOR:
Enrique F. Larreta
REDACTOR JEFE:
Emiliano Juárez
REDACCION:
Ignacio Borrell, Octavio López,
Antonio del Rio
DISEÑO:
Esteban Pérez

Editado por PUBLINFORMATICA, S. A.
Bravo Murillo, 377. 5.º A. Tel.:
733 74 13 - 28020 Madrid
Presidente:
Fernando Bolín
Director Editorial Revistas de Usuarios:
Juan Arencibia
Director de Ventas:
Antonio González
Producción: Miguel Onieva
Servicio al cliente:
Julia González. Tel.: 733 79 69
Administración:
PUBLINFORMATICA, S. A.
Publicidad Madrid:
Emilio García
Dirección, Publicidad y Administración:
Bravo Murillo, 377. 5.º A. Tel. 733 74 13.
Télex: 48877 OPZX e 28020 Madrid
Publicidad Barcelona:
Lidia Cendrós. Pelayo, 12. Tels. (93)
318 02 89 - 301 47 00, ext. 27 y 28.
08001 Barcelona
Depósito legal: M-29041-1984
Distribuye S.G.E.L. Avda. Valdelaparra,
s/n. Alcobendas (Madrid).
Fotomecánica: Karmat, C/ Pantoja, 10.
Madrid.
Fotocomposición: Espacio y Punto
Imprime: Héroes, C/ Torrelara, 8. Madrid.
Distribuidor en VENEZUELA,
SIPAM, S. A.
AVD. REPUBLICA DOMINICANA, EDIF.
FELTREC - OFICINA 4B BOLEITA SUR
CARACAS (VENEZUELA)
Esta publicación es miembro de la
Asociación de Revistas de
Información ari asociada a la
Federación Internacional de Prensa
Periódica, FIPP.

SUSCRIPCIONES:
Rogamos dirijan toda la correspondencia
relacionada con suscripciones a:
TODOSPECTRUM EDISA: Tel. 415 97 12
C/ López de Hoyos, 141-5º
28002 MADRID
(Para todos los pagos reseñar solamentes
TODOSPECTRUM)
Para la compra de ejemplares atrasados
diríjanse a la propia editorial
TODOSPECTRUM
C/ Bravo Murillo, 377. 5.º A
Tel. 733 74 13 - 28020 MADRID

Si deseas colaborar en TODOSPECTRUM
remite tus artículos o programas a Bravo
Murillo, 377. 5.º A. 28020 Madrid. Los
programas deberán estar grabados en
cassette y los artículos mecanografiados.
A efectos de remuneración, se analiza cada
colaboración aisladamente, estudiando su
complejidad y calidad.

EDITORIAL

NOVEDADES EN SONIMAG

Por fin hay fecha definitiva para el lanzamiento del nuevo ordenador Sinclair construido por Amstrad. El Salón Internacional de la Imagen, el Sonido y la Electrónica, SONIMAG, que se celebrará en Barcelona del 15 al 21 de este mes, será el escenario de la presentación del Spectrum + 2, del que esperamos poder ofreceros nuestras primeras impresiones en el próximo número de Todospectrum.

Durante el SONIMAG hará también su primera aparición en público el Amstrad PC, compatible IBM con el que la empresa de Alan Sugar se incorpora a la tendencia imperante en el mercado de ordenadores personales.

Por otra parte, son noticia las recientes declaraciones de Sir Clive Sinclair a la prensa especializada del Reino Unido, en las que aporta algunos datos sobre el pasado reciente de Sinclair Research. «Si Alan Sugar puede obtener beneficios de él –afirmó refiriéndose al Spectrum– pues fantástico, pero nosotros estábamos perdiendo dinero y eso no tenía sentido». Sir Clive achacó el escaso éxito comercial del QL a la utilización del microprocesador 68008: «Yo deseaba realizar el proyecto que originó el QL partiendo del Z-80, pero Nigel Searle y parte de los ingenieros quisieron desarrollarlo basándose en el 68000. Lo cierto es que no había nada que pudiera hacerse con el 68000 y no con el Z-80».

Y puesto que hablamos del QL, deciros que, como ya hicimos en agosto del año pasado, hemos decidido dar unas cortas vacaciones (hasta noviembre) a nuestro habitual Suplemento QL.





ALIEN HIGHWAY

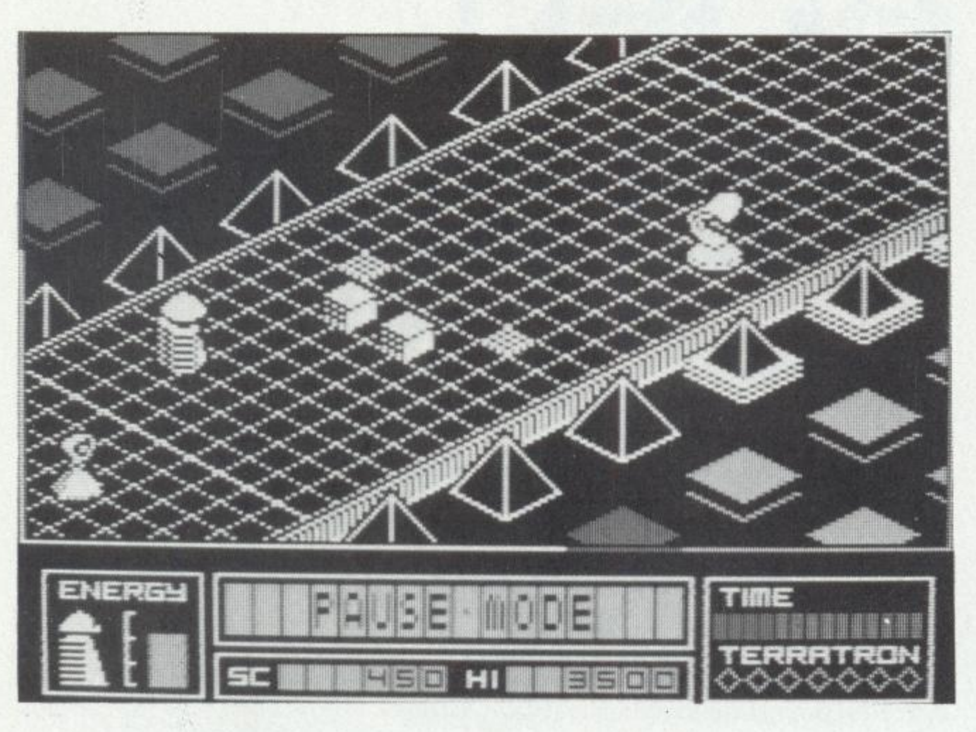
En esta ocasión nos enfrentamos en la guía de Hackers con el programa Alien Highway. Se trata de la segunda parte del conocido Highway Encounter. El juego en sí es muy parecido a su predecesor, y las diferencias más importantes son que en esta ocasión disponemos de un único «Vorton» cuya energía va disminuyendo cada vez que somos alcanzados por un alienígena, y que las pantallas se suceden en un orden aleatorio. El objetivo del juego es bastante simple, hay que salvar la tierra de la dominación de los invasores. Para ello disponemos de un Terratron, que parece que es el único arma con el que podremos conseguirlo. Tendremos que ir empujándolo a lo largo de una autopista e ir cargándolo durante el camino en las siete estaciones regeneradoras existentes.

Lo primero que llama la atención a un hacker que pretende adentrarse en este programa es que después de un par de bloques normales aparecen los clásicos tonos de sincronismo de los turbos pulsados que popularizó en otra época Speedlock. A la vista de esto es posible que la mayor parte de este artículo se centre en la lucha contra la protección. Pero no hay que adelantarse.

Empezaremos como siempre por lo más sencillo, cargamos el primer bloque y pulsamos Break. El programa se para. Intentamos listar y aparece el clásico perfil del listado con tinta y papel del mismo color. Cambiamos el color de la tinta para poder ver el primer número de línea y nos encontramos con que como cabía esperar tiene el número cero. Para cambiarlo hay que pokear el segundo byte del área del programa BASIC. La dirección absoluta depende de los periféricos que estén conectados. Si queremos referirlo a la variable PROG deberemos hacer POKE PEEK 23635+256*PEEK 23636+1,1. Con esto ya podemos editar la línea. Lo hacemos y con mucho cuidado de no volver a introducir la línea vamos borrando los códigos de cambio de color para poder ver el listado de la línea.

Sorprendentemente contiene una llamada USR con un número de siete cifras. Evidentemente tiene que estar cambiado, y para averiguar el verdadero valor tendremos que mirar cuál es el número codificado a continuación. Borramos del todo la línea que estamos editando (si pulsáramos ENTER volveríamos a introducirla en el programa destruyendo el auténtico valor) y cargamos el monitor.

Comenzamos a analizar con cuidado la codificación del programa. Nos encontramos con los cambios de color, los códigos de las instrucciones BASIC y al llegar al USR están los códigos ASCII del número de siete cifras, el dato #OE que indica el comienzo de la verdadera codificación del número y a continuación los cinco bytes que nos interesan. Pero ni siquiera es la codificación correspondiente a un número entero. Podríamos intentar decodificarlo a mano, pero lo más cómodo es modificar el programa desde el monitor para que en vez de llamar a esa dirección la escriba en la pantalla. Hacemos esto y el resultado es que la dirección es 23829.1. El hecho de que no sea en-



tero es solamente para dificultar un poco más su interpretación.

Pasamos a esta dirección teniendo en cuenta que el interface 1 desplaza todo el área BASIC unos cuantos bytes hacia delante. Nada más comenzar a desensamblar nos encontramos con algo que no parece un programa. Está lleno de códigos #DD que el MONS desensambla como NOP, pero que sabemos que corresponden a una forma de manipular el registro IX como si se tratara de dos registros de ocho bits. La forma de hacerlo es colocando el código #DD y a continuación la operación que queremos hacer con el registro IX pero referida a HL. Así si ponemos LD A,L lo que hace en realidad es cargar en el acumulador el byte menos significativo del registro IX. Además nos encontramos con muchas instrucciones que no tienen sentido como LD B,B etc. La principal característica de este tipo de programas es que están escritos para que no se entiendan. Nos armamos de paciencia y seguimos lo que hace paso a paso. Parece que lo más importante es que coloca la pila en #FFF8 y luego la reconstruye para volver al BASIC. Para comprobar si realmente esto es lo único que importa probamos a borrar toda la memoria, hacer un CLEAR en la dirección 65535 y cargar el programa saltándonos el primer bloque. Todo funciona perfectamente, de lo que se deduce que este primer bloque sólo sirve para despistar.

El primer bloque del programa contiene una llamada USR con un número de siete cifras.

Pasamos al segundo bloque. Este ya no se puede parar de una manera tan simple como el anterior. El pulsar Break no sirve para nada, y está preparado para que el Merge no funcione. Habrá que utilizar métodos más drásticos. En estos casos mi sistema preferido es el siguiente: primero se carga el monitor, se hace un LOAD"" y se lee solamente la cabecera. Con esto se consigue que las variables del sistema tomen los valores que indican la longitud del programa. Por último entramos en el monitor y escribimos un programita que carge un único bloque de datos sin cabecera a partir de la dirección de comienzo de la memoria de programa, de esta manera conseguimos tener en memoria el programa exactamente como si se hubiese grabado sin autoejecución.

Este segundo programa luce en su primera línea el nombre de la protección. Es Speedlock. Este es uno de los sistemas más antiguos y que más veces ha sido utilizado. Por esta razón, a no ser que hayan cambiado algo, no ofrecerá muchos problemas el llegar hasta el juego. De todas formas explicaremos paso a paso cómo funciona.

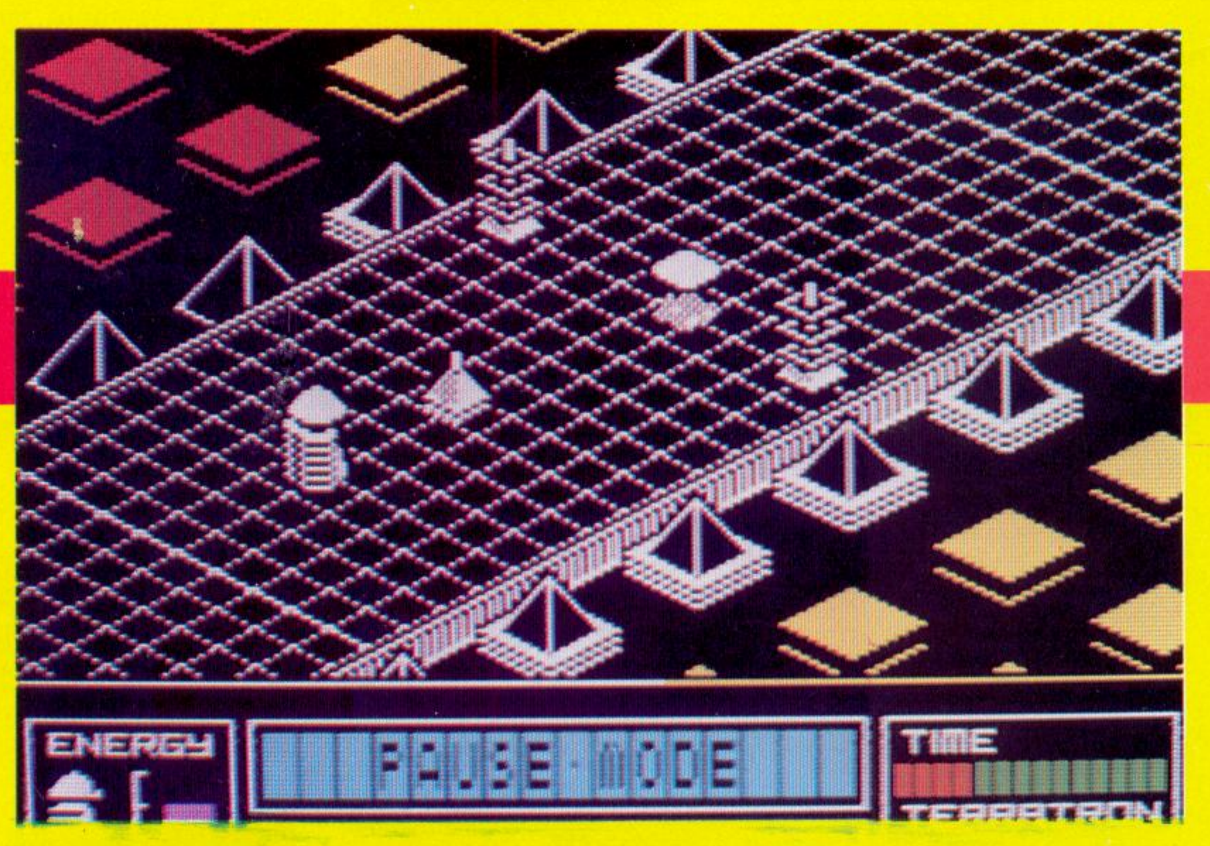
Lo primero es el listado BASIC. Está lleno de caracteres de cambio de color que dificultan el listado. Además todas las líneas tienen el número cero, con lo que no podremos editarlas a no ser que las pokeemos antes. Por esto es mejor analizarlo desde el monitor. Nos encontramos con que lo único que hace es un montón de POKEs en las variables del sistema en los que por supuesto no coincide el número codificado en ASCII con el auténtico valor del dato que está detrás del dato #OE. lo que realmente hace es POKEar en la pila la

```
  10 CLEAR 64255: LET n=64256: R
ESTORE
  30 FOR i=200 TO 240 STEP 10
  40 READ a$,a: LET s=0
  50 FOR j=1 TO LEN a$-1 STEP 2
  60 LET d=16*(CODE a$(j)-48-7*(
a$(j)>"9"))+CODE a$(j+1)-48-7*(a
$(j+1)>"9")
  70 POKE n,d: LET n=n+1: LET s=
s+d: NEXT j
  80 IF s<>a THEN  PRINT "Error
en la linea ";i: STOP
  90 NEXT i
 200 DATA "2A555CED5B535CCDE519D
D21057D119A063EFF37CD560530F1F32
1377F11ABFC",3597
 210 DATA "01F6023EC8ED4FCD6BFB2
1D9FC545D01C8023E82ED4FCD6BFB21A
BFC11ABEC01",4224
 220 DATA "9000EDB021E1FC36EC3EC
93225FDCDD9FC215FFB227DFF3E61329
9FFC33CFFFB",4805
 230 DATA "FD213A5CED7B3D5CC3B31
BCD5200ED5FAE77EDA0E03B3BE8F3210
06111005B01",3715
 240 DATA "0099EDB0C3AB87",1067
 300 RANDOMIZE USR 64256
 310 CLEAR 64255: INPUT "Inmorta
l ?";a$: IF a$="s" THEN  POKE 40
947,201
 320 INPUT "Tiempo infinito ?";a
$: IF a$="s" THEN  POKE 36661,0
 330 INPUT "Apoyarse en los late
rales ?";a$: IF a$="s" THEN  POK
E 39331,165
 340 INPUT "Mapa fijo ?";a$: IF
a$="s" THEN  POKE 45157,201
 350 RANDOMIZE USR 64376
```

dirección a la que debe volver dentro del bucle principal del BASIC cuando acabe la ejecución o se produzca un error. De esta forma, en lugar de volver al BASIC, salta a un programa en código máquina que comienza en la dirección que se ha POKEado y que ha sido extraída de la variable del sistema VARS. Esta dirección es #622D si está conectado el interface 1. Aquí nos encontramos con el clásico programa en código máquina de este sistema de protección. Es totalmente ininteligible. Parece como si estuviéramos desensamblando una tabla de datos en lugar de un programa. La mayor parte de las instrucciones no valen para nada y las que sirven hay que buscarlas con mucho cuidado. En grandes rasgos lo que hace es desactivar las interrupciones, carga un dato en el registro R, realiza un montón de instrucciones que no sirven para nada entre las que se incluye un LDIR que mueve casi toda la memoria libre, y por último entra en un bucle en el que desenmascara un bloque del programa utilizando el registro R y lo coloca a partir de la dirección #FCD9. Este bucle es muy curioso ya que no tiene ninguna instrucción de salto. En su lugar se utiliza la instrucción RET PE después de haber colocado la pila de forma que el salto se produzca a la dirección deseada.

El segundo programa luce en su primera línea el nombre de la protección: Speedlock.

El hecho de que la máscara se haga con el registro R implica que no podemos modificar el trozo de programa comprendido entre el punto en que se carga este registro y el bucle en que se utiliza, ya que se incrementa cada vez que se ejecuta una instrucción. Esto impide que eliminamos el



LDIR que modifica toda la memoria y que destruiría cualquier monitor existente. No tenemos más remedio que escribir un pequeño programa que deshaga la máscara, pero necesitamos conocer el valor del registro R la primera vez que se ejecuta el bucle. En este tipo de situaciones en que queremos saber un dato, pero cuando todo el sistema está destrozado, se puede utilizar el siguiente truco. Justo en el punto en que está el dato que buscamos insertamos unas instrucciones con las que guardamos el dato en una dirección alta de memoria, colocamos en la variable del sistema RAMTOP una dirección menor que aquella en la que tenemos el dato, y por último hacemos un salto a la dirección de NEW (#11B7). Esto reinicializará el sistema, pero sólo borrará hasta el RAMTOP, con lo que podremos leer el dato que queríamos directamente de la memoria con la función PEEK. Haciendo esto descubrimos que el primer valor que se utiliza como máscara es el #CD. Con esto ya podemos deshacer la primera máscara y pasar el análisis a la dirección #FCD9.

Aquí nos encontramos con otra máscara similar pero conservando el antiguo valor del registro R. Para deshacer esta segunda parte utilizamos un método similar al anterior, pero en esta ocasión dejamos que la máscara se pase directamente y después el NEW nos dejará a salvo toda la rutina que queda. Hasta aquí todo es igual que las antiguas versiones de este sistema de protección, pero en este punto nos encontramos ya con la rutina de carga. En esta ocasión todavía queda por superar una última barrera en la que se va pasando una máscara encadenada varias veces en las que se utilizan como valores iniciales los propios códigos del programa que se está ejecutando. De esta forma si cambiamos un solo byte del programa no conseguiremos llegar a la rutina de carga, y si no lo cambiamos, la rutina se ejecutará impidiendo que tomemos el control. La solución para superar esta última parte consiste en copiar el programa en otra dirección y utilizar esta copia para coger los valores iniciales mientras que en la otra se pueden colocar puntos de ruptura para evitar que se lance el proceso de carga. Con esto ya hemos llegado hasta la auténtica rutina, aunque todavía realiza una última comprobación antes de ejecutarla. Suma todos los bytes de una zona de la rutina, y si no coincide con lo que debería de dar cuelga el programa. Realmente esta comprobación no tiene mucho sentido ya que si hemos logrado superar todas las protecciones anteriores ésta debe de dar el resultado correcto, y si no no hay nada que ejecutar.

El cargador es prácticamente igual al que han utilizado siempre, con la clásica medida de períodos en el espacio comprendido entre la simulación de cabecera y el bloque principal. (Con esto se pretende evitar las copias analógicas hechas con casetes de nivel de grabación automático).

GUIA DEL HACKER

El bloque de datos que carga ocupa toda la memoria comprendida entre #4000 y #F408, donde se incluye al principio la pantalla de presentación que aparece durante la carga y justo al final la que aparece cuando acaba. Después el programa se lanza en #87AB.

Con todo esto ya estamos preparados para comenzar el análisis del juego, pero estamos llegando al final del artículo, por lo que nos limitaremos a dar unas pocas indicaciones junto con los POKEs fundamentales.

El mapa no está codificado como tal, sino que tiene una tabla a partir de la dirección #7100 donde está codificada la información de los 240 objetos que hay. Para cada uno tiene 16 bytes que son directamente los que utiliza durante el juego para controlar su movimiento. Estos objetos pueden ser cualquier de los que aparecen en el juego, tanto fijos como móviles. La tabla en total ocupa 3.75 Kbytes y la rutina en la dirección #89D9 se encarga de cambiar el orden de las pantallas en cada partida. Neutralizándola conseguiremos jugar siempre con el mismo mapa.

> La rutina de la dirección #9292 se encarga de gestionar el movimiento de todos los objetos.

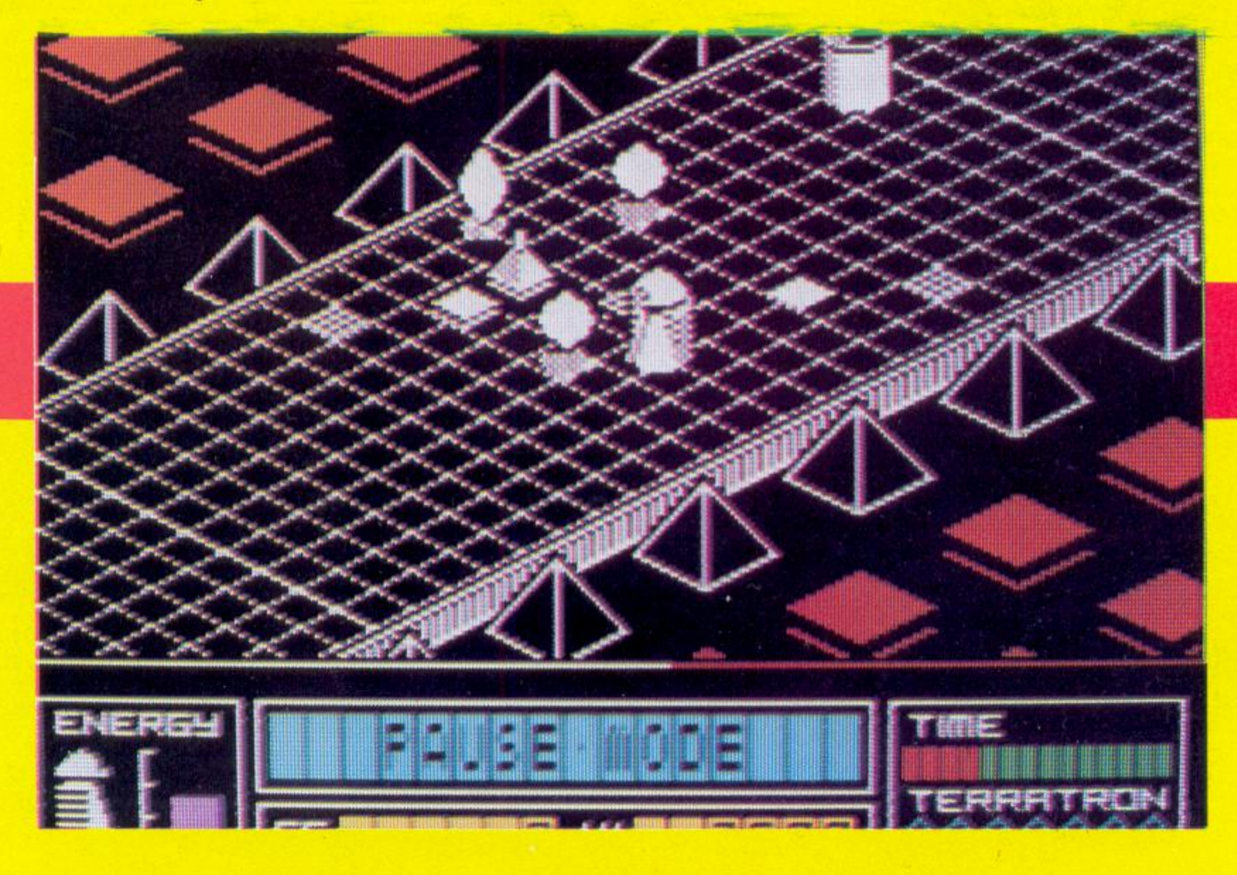
La rutina en la dirección #9292 se encarga de gestionar el movimiento de todos los objetos, lo que incluye a nuestro Vorton, el Terratron y los tres disparos. La gestión de todos ellos está muy mezclada por lo que resulta difícil hacer POKEs que sólo afecten a nuestros enemigos. Con el programa 1 podréis jugar siendo inmortales y otras opciones. También lo podéis utilizar para probar otros POKEs colocándolos a partir de la línes 301. A modo de ejemplo proponemos estos dos:

POKE 39330,195
POKE 39090,180

Si intentáis buscar más POKEs tened en cuenta que el cargador coloca el programa desplazado 1536 bytes respecto a su dirección de funcionamiento.

Por último hay que comentar que existe una última comprobación de protección en el programa. Durante el proceso de carga se ha manipulado el registro R y se ha colocado en él un dato con el bit 7 puesto a uno. Al ir incrementando el contenido para refrescar las memorias se hace solamente sobre siete bits, permaneciendo inalterado el más significativo. Este bit se comprueba constantemente durante el juego utilizando una curiosa instrucción de rotación no estándar cuyos códigos son #CB #37. Su efecto parece ser una rotación a la izquierda del acumulador pasando el último bit al carry e introduciendo un uno por la derecha. Probablemente se habrá utilizado esta instrucción por el hecho de que los monitores no la desensamblan.

Manuel Arana





SERVICIO DE

Completa tu colección de ZX.
A continuación te resumimos el contenido de los ejemplares atrasados en existencia.



Núm. 3/300 ptas.
El Spectrum por dentro. Quince programas, juegos y montajes Software.



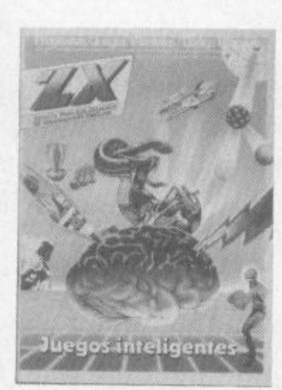
Núm. 4/300 ptas.
QL, el nuevo Sinclair. Dieciocho programas, juegos, montajes, ideas/Novedades.



Núm. 5/300 ptas.
Gráficos y sonido en el Spectrum/Libros/Software/ 13 programas.



Núm. 6/300 ptas.
Construya su propio juego/13 programas y montajes/ideas/Software.



Núm. 7/300 ptas.
Juegos inteligentes/Software/ 11 programas/Libros.



Núm. 8/300 ptas.
La aventura es la aventura/12 programas/Juegos y montajes/Código máquina.



Núm. 9/300 ptas.
Construye tu propio juego. Catorce programas para el verano. Gráficos en el Spectrum.



Núm. 10/300 ptas.
Catorce programas educativos: geografía, cramer, gráficos, razones trigonométricas, elongación. Código máquina.



Núm. 11/300 ptas.
Cómo crear marcianos y otros monstruos.
Diez programas satélites de júpiter, rescate, interés, circulo, préstamo hipotecario.



Núm. 12/300 ptas.
Presentación del Spectrum Plus. Forth, capítulo 1. Gráficos en el Spectrum, 4 parte. Libros. Programas y montajes.



Núm. 13/300 ptas.
Guía del software para el Spectrum todos los programas del mercado. Forth, capítulo 2. Visitamos Sinclair Research. Libros. Programas.



Núm. 14/300 ptas.
Cómo jugar al Hobbit. Gráficos de funciones. Programas de ajedrez. Conexiones con el P I/O. Programas Multiplic, enseñar deleitando. Libros, Forth, tercera parte.



Núm. 15/300 ptas.
Simuladores de vuelo. Forth, cuarta parte. Montajes: Reloj digital para Spectrum. BASIC para principiantes. Libros. Programas.



Núm. 16/300 ptas.
Cassettes: solución a los problemas de grabación. Test de Psicología. Sistema de Desarrollo para el ZX-81. Cinemática. Programas. Animación Gráfica. BASIC para principiantes (2). Forth, quinta parte.



Núm. 17/300 ptas.
Mapa de Atic-Atac. Estira de caracteres. Dinámica de una partícula. Libros. QL Magazine. Programas. Convertidor analógico-digital con el P I/O.

EJEMPLARES ATRASADOS



Núm. 18/300 ptas.
Rentas 85. Forth, sexta parte. Programas. BASIC para principiantes (3). Plotting Gráficos. Libros. Usuarios. Crítica.



Núm. 19/300 ptas.
Mapa de Knight Lore. Noticias. Crítica. Renta 85 (segunda parte). Libros. El ZX-81 aprende a sumar. Scroll de ventanas. Programas. El software que nos invade. BASIC para principiantes (4).



Núm. 20/300 ptas.
Vacaciones con informática. Crítica. Noticias. Programas. Son muy divertidos. Libros. Generación de placas de circuito impreso. Forth. Movimiento armónico simple. Spectrum musical.



Núm. 21/300 ptas.
Mapa de Underwurlde. Noticias. Crítica. ¿Has probado? Programa especial: barquitos. Sois muy divertidos. Libros para el verano. Un poco de física. BASIC para principiantes (5).



Núm. 22/300 ptas.
Noticias. Teclados profesionales. Crítica. ¿Has probado? Programa especial: procesador de textos. Generación de placas de circuito impreso (segunda parte). Programas QL español. Quinielas en Spectrum. BASIC para principiantes (6).



Núm. 23/300 ptas.
Crítica. ¿Has probado? Profanation profanado. Noticias. Discos para Spectrum. Dossier educación: Spectrum en el aula, autoevaluación, Logo. Código máquina. Programación especial: quinielas. Montaje a cámara lenta. BASIC para principiantes (7).



Núm. 24/300 ptas.
Juegos/Mapas del Nodes of Yesod y Lords of Midnight/¿Has probado?/Sois muy divertidos/Usuario/Ajuste de gráficas/Multisearch/Programas/Montaje: inversor de vídeo para ZX 81/Dossier QL.



Núm. 25/300 ptas.
Juegos/Especial juegos, Mapas y trucos de: Highway encounter, Tir Na Nog, Nightshade/¿Qué es el Stack?/Programa especial/Código máquina/Lotería primitiva/Stándares de la informática/Programas.



Núm. 26/300 ptas.
Spectrum o QL, invasión de los 128/¿Cómo utilizar mejor el microdrive?/Juegos/Mapa del Dun Darach y misión imposible/Programación estructurada/BASIC.



Núm. 27/300 ptas.
La vida de Sinclair/Piezas musicales para Spectrum/Juegos/Mapas del ARNHCM y SABOTEUR/Areas/BASIC para impresora/El área de variable y la instrucción RST 16.


DISPONEMOS DE TAPAS ESPECIALES PARA SUS EJEMPLARES DE ZX (sin necesidad de encuadernación)

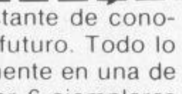


PRECIO UNIDAD **650** ptas.

(en cada tomo se pueden encuadernar 6 números)

Para hacer tu pedido, rellena el cupón adjunto, córtalo y envíalo HOY MISMO a:

ZX, Bravo Murillo, 377 • 28020-MADRID • Tel. 733 74 13



Los ejemplares atrasados de ZX serán una fuente constante de conocimientos, ideas, soluciones y entretenimientos para el futuro. Todo lo anterior hace recomendable que los guardes ordenadamente en una de las tapas especiales para ZX. Cada tapa puede contener 6 ejemplares y cuesta solamente 650 ptas.

Ruego me envíen los siguientes ejemplares atrasados de ZX .... al precio de 300 ptas. cada uno

Por favor envíen .... tapa(s) al precio de 650 ptas. cada una (+ gastos de envío).

El importe lo abonaré:
☐ contra reembolso ☐ cheque adjunto ☐ con mi tarjeta de crédito
☐ American Expres ☐ Visa ☐ Interbank.

Fecha de caducidad ....

Número de mi tarjeta ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

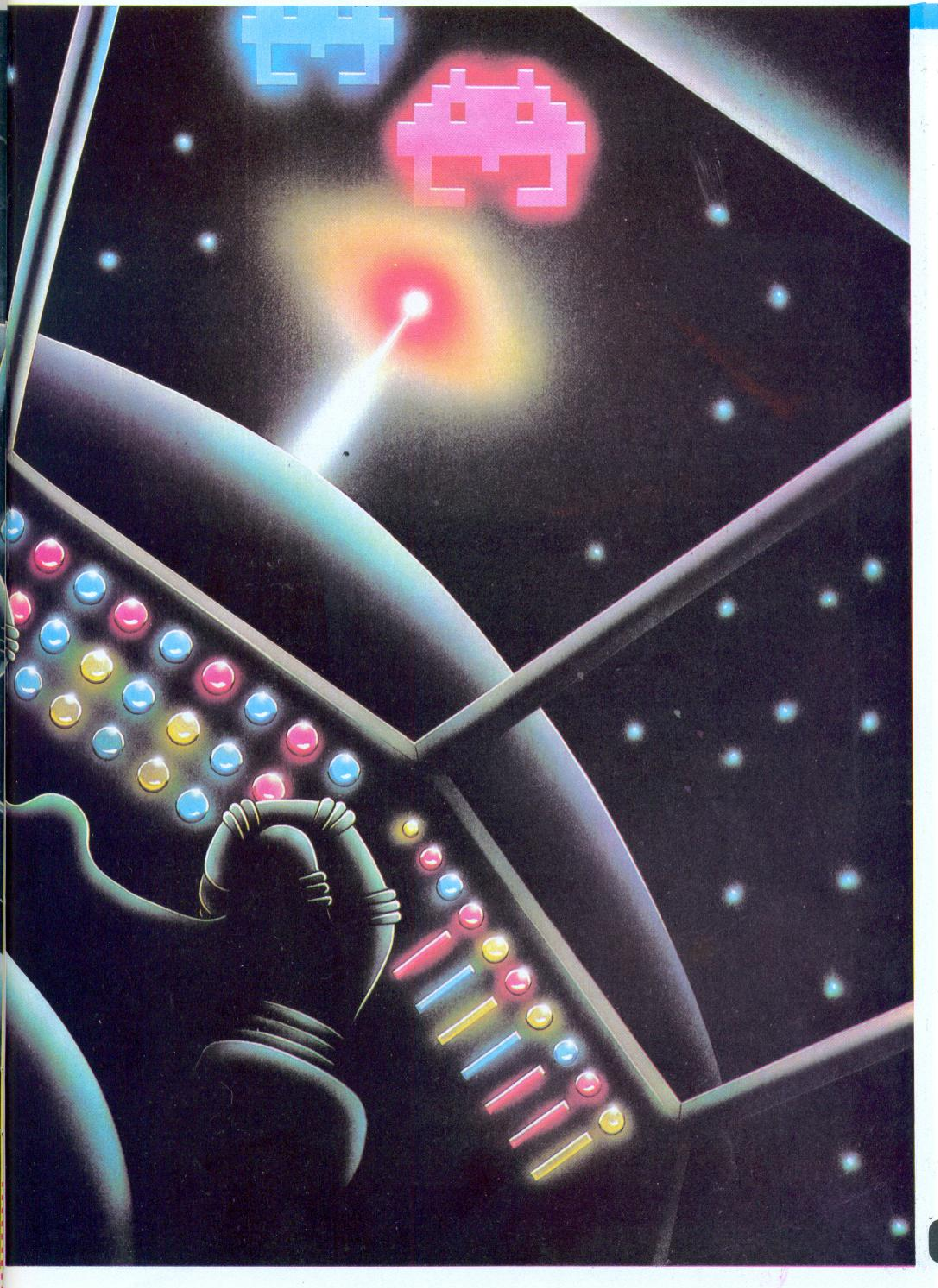
NOMBRE ....
DIRECCION ....
POBLACION .... C.P. ....
PROVINCIA ....

DISEÑO de SPRITES

El Spectrum ofrece grandes posibilidades a la hora de crear gráficos, pero ahí queda la cosa. A diferencia de otros ordenadores, no lleva implementada en su sistema operativo ninguna rutina destinada a la creación de sprites, formas que pueden desplazarse a través de la pantalla de manera independiente, pudiendo especificarse para cada una de ellas la velocidad y dirección del movimiento. Son muy efectivos cuando se trata de diseñar juegos tipo Arcade debido a la facilidad de su manejo.

Existen múltiples formas de crear animación para su visualización en la pantalla. El más obvio consiste en la creación de imágenes fijas almacenadas en la memoria, las cuales se suceden unas detrás de otra, con lo cual se da la impresión de movimiento. Esta técnica es similar a la utilizada por el cine. Sin embargo, tiene el inconveniente de que necesita una gran cantidad de memoria para poder almacenar las diferentes pantallas, y todos sabemos que los microordenadores como el Spectrum sólo puede realizar secuencias animadas de este tipo hasta cierto punto debido a la escasa memoria disponible. De todas formas, siempre existen trucos de programación que permiten aumentar la capacidad de almacenamiento (utilizar una porción de la pantalla, compresión de datos etc.). Por supuesto este tipo de animación sólo puede conseguirse eficazmente mediante el empleo de rutinas en código máquina.

La forma más usual de crear movimiento consiste en imprimir un determinado carácter, borrarlo (imprimiendo un espacio en blanco sobre él), y volverlo a imprimir en una posición de pantalla algo más alejada. Con el empleo del BASIC sólo puede conseguirse movimiento en baja rseolución (carácter a carácter o de 8 en 8 pixels), pero resulta bastante lento y poco efectivo. Utilizan-



do el lenguaje ensamblador, puede conseguirse también movimiento en alta resolución (pixel a pixel), pero ello requiere una gran práctica de programación para evitar parpadeos en la imagen de televisión, y a medida que se van añadiendo más objetos, los problemas para controlarlos individualmente aumentan de forma considerable.

La técnica más avanzada de movimiento de figuras es la que se conoce con el nombre de gráficos sprites, que es la que normalmente se emplea para la creación de los juegos de las máquinas de los recreativos.

La técnica más avanzada de movimiento de figuras es la que se conoce con el nombre de gráficos sprites.

Sprites

El término sprite, traducido literalmente, ya es de por sí muy expresivo, pues significa duende o trasgo. Los sprites normalizados que utilizan ordenadores como el Commodore 64, además de poderles especificar la velocidad y dirección, tienen capacidad de moverse en diferentes planos superpuestos (unos sprites pueden pasar a través de los otros). Cada sprite tiene un número asignado. Por ello, si dos sprites coinciden en una misma posición de pantalla, sólo se visualizará el que tiene el número más bajo.

También puede colorearse individualmente cada sprite e incluso reducirlo de tamaño (variando simplemente una determinada dirección de memoria). Otra condición que no debe faltar en un sprite que

```
  10         ORG   #80F5
  20         ENT   $
  30 ON      LD    A,#80
  40         LD    I,A
  50         IM    2
  60         RET
  70 OFF     IM    0
  80         RET
  90 IADDR   DEFW  #8101
 100 SPRITE  PUSH  AF
 110         PUSH  BC
 120         PUSH  DE
 130         PUSH  HL
 140         PUSH  IX
 150         LD    IX,(#5C4B)
 160         LD    A,(IX+0)
 170         CP    #D3
 180         JR    NZ,EXIT
 190         LD    B,(IX+4)
 200 SPLOOP  PUSH  BC
 210         LD    BC,8
 220         ADD   IX,BC
 230         CALL  NXTSPR
 240         POP   BC
 250         DJNZ  SPLOOP
 260 EXIT    POP   IX
 270         POP   HL
 280         POP   DE
 290         POP   BC
 300         POP   AF
 310         RST   #38
 320         RET
 330 LINE    RRC   H
 340         RRC   H
 350         RRC   H
 360         LD    BC,#1F
 370         ADD   HL,BC
 380         RLC   H
 390         RLC   H
 400         RLC   H
 410         RET
 420 WIPE    CALL  HWIPE
 430         CALL  LINE
 440 HWIPE   CALL  QWIPE
 450         INC   HL
 460 QWIPE   PUSH  HL
 470         LD    B,8
 480 WPLOOP  LD    (HL),0
 490         INC   H
 500         DJNZ  WPLOOP
 510         POP   HL
 520         RET
 530 TEST    XOR   A
 540         CALL  HTEST
 550         CALL  LINE
 560 HTEST   CALL  QTEST
 570         INC   HL
 580 QTEST   PUSH  HL
 590         LD    B,8
 600 TSLOOP  OR    (HL)
 610         INC   H
 620         DJNZ  TSLOOP
 630         POP   HL
 640         RET
 650 FINDA   LD    A,B
 660         AND   #18
 670         OR    #40
 680         LD    H,A
 690         LD    A,B
 700         RRCA
 710         RRCA
 720         RRCA
 730         AND   #E0
 740         OR    C
 750         LD    L,A
 760         RET
 770 NXTSPR  LD    A,(IX+7)
 780         CP    #40
 790         RET   C
 800         DEC   (IX+1)
 810         RET   NZ
 820         LD    A,(IX+2)
 830         LD    (IX+1),A
 840         LD    B,(IX+3)
 850         LD    C,(IX+4)
 860         CALL  FINDA
 870         CALL  WIPE
 880         LD    A,(IX+3)
 890         ADD   A,(IX+5)
 900         CP    #17
 910         JR    NC,NSEXIT
 920         LD    B,A
 930         LD    A,(IX+4)
 940         ADD   A,(IX+6)
 950         CP    #1F
 960         JR    NC,NSEXIT
 970         LD    C,A
 980         PUSH  BC
 990         CALL  FINDA
1000         CALL  TEST
1010         POP   BC
1020         AND   A
1030         JR    NZ,NSEXIT
1040         LD    (IX+3),B
1050         LD    (IX+4),C
1060         JR    NSDRAW
1070 NSEXIT  LD    (IX+07),0
1080         LD    B,(IX+3)
1090         LD    C,(IX+4)
1100 NSDRAW  LD    L,(IX+0)
1110         DEC   L
1120         LD    H,0
1130         ADD   HL,HL
1140         ADD   HL,HL
1150         ADD   HL,HL
1160         ADD   HL,HL
1170         ADD   HL,HL
1180         LD    DE,(#5C7B)
1190         ADD   HL,DE
1200         PUSH  HL
1210         CALL  FINDA
1220         POP   DE
1230 DRAW    CALL  HDRAW
1240         CALL  LINE
1250 HDRAW   CALL  QDRAW
1260         INC   HL
1270 QDRAW   PUSH  HL
1280         LD    B,8
1290 DRLOOP  LD    A,(DE)
1300         LD    (HL),A
1310         INC   DE
1320         INC   H
1330         DJNZ  DRLOOP
1340         POP   HL
1350         RET
```

```
  10 CLEAR 33012: DIM s$(2,8)
  20 FOR j=1 TO 2: FOR i=1 TO 8:
 READ a: LET s$(j,i)=CHR$ a: NEX
T i: NEXT j
  30 FOR j=0 TO 31: READ a: POKE
 USR "a"+j,a: NEXT j
 500 LET t=0: FOR i=33013  TO 33
257: READ a: POKE i,a: LET t=PEE
K i+t: NEXT i: IF t<>29400 THEN
 PRINT "error en datas",t: STOP
 520 RANDOMIZE USR 33013
 535 LET fil=CODE s$(1,4): LET c
ol=CODE s$(1,5): LET f=CODE s$(2
,4): LET c=CODE s$(2,5)
 540 IF fil=0 OR fil=22 THEN  LE
T s$(1,6)=CHR$ (256-CODE s$(1,6)
): BEEP .03,24
 545 IF col=0 OR col=30 THEN  LE
T s$(1,7)=CHR$ (256-CODE s$(1,7)
): BEEP .03,12
 550 IF f=0 OR f=22 THEN  LET s$
(2,6)=CHR$ (256-CODE s$(2,6)): B
EEP .03,24
 555 IF c=0 OR c=30 THEN  LET s$
(2,7)=CHR$ (256-CODE s$(2,7)): B
EEP .03,12
 600 LET s$(1,8)=CHR$ 65: LET s$
(2,8)=CHR$ 65
 700 GO TO 535
 900 REM datos de los sprites
 910 DATA 1,1,2,11,6,255,1,65
 920 DATA 1,1,2,11,20,1,255,65
 930 REM configuracion sprite
 940 DATA 7,24,32,64,64,128,128,
128
 950 DATA 224,24,4,2,2,1,1,1
 960 DATA 128,128,128,64,64,32,2
4,7
 970 DATA 1,1,1,2,2,4,24,224
 990 REM codigo maquina
1000 DATA 62,128,237,71,237,94,2
01,237,70,201
1001 DATA 1,129,245,197,213,229,
221,229,221,42
1003 DATA 75,92,221,126,0,254,21
1,32,15,221
1004 DATA 70,4,197,1,8,0,221,9,2
05,113
1005 DATA 129,193,16,244,221,225
,225,209,193,241
1006 DATA 255,201,203,12,203,12,
203,12,1,31
1007 DATA 0,9,203,4,203,4,203,4,
201,205
1008 DATA 64,129,205,41,129,205,
68,129,35,229
1009 DATA 6,8,54,0,36,16,251,225
,201,175
1010 DATA 205,85,129,205,41,129,
205,89,129,35
1011 DATA 229,6,8,182,36,16,252,
225,201,120
1012 DATA 230,24,246,64,103,120,
15,15,15,230
1013 DATA 224,177,111,201,221,12
6,7,254,64,216
1014 DATA 221,53,1,192,221,126,2
,221,119,1
1015 DATA 221,70,3,221,78,4,205,
98,129,205
1016 DATA 58,129,221,126,3,221,1
34,5,254,23
1017 DATA 48,31,71,221,126,4,221
,134,6,254
1018 DATA 31,48,20,79,197,205,98
,129,205,78
1019 DATA 129,193,167,32,8,221,1
12,3,221,113
1020 DATA 4,24,10,221,54,7,0,221
,70,3
1021 DATA 221,78,4,221,110,0,45,
38,0,41
1022 DATA 41,41,41,41,237,91,123
,92,25,229
1023 DATA 205,98,129,209,205,219
,129,205,41,129
1024 DATA 205,223,129,35,229,6,8
,26,119,19
1025  DATA 36,16,250,225,201
```

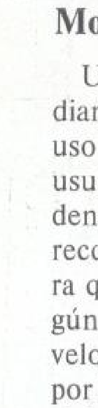
se precie de serlo es la detección de colisión.

Los gráficos sprite tienen la ventaja de que se les puede controlar desde el BASIC sin perder apenas efectividad en lo que a velocidad se refiere.

Duendes en el Spectrum

El Spectrum no dispone de gráficos sprites, pero con la programación adecuada pueden simularse perfectamente. La rutina que vamos a comentar en este artículo permite la creación de sprites con ciertas limitaciones. No permite la asignación de colores ni los planos superpuestos (no puede pasar un sprite detrás de otro) y el movimiento se lleva a cabo en modo de baja resolución, pero sí podemos especificar la velocidad y dirección del movimiento, detectándose las colisiones de forma automática.

El programa hace posible que puedan evolucionar en la pantalla hasta un total de 5 sprites diferentes de 16 pixels de ancho por 16 de alto (4 caracteres para cada sprite). El número máximo de sprites que puedan visualizarse a la vez sólo vendrá limitado por las dimensiones de la pantalla.



Moviéndose a toda velocidad

Una vez creado el sprite mediante el método habitual, haciendo uso de los gráficos definidos por el usuario, se deben asignar las coordenadas iniciales, su velocidad y dirección. Todo esto es suficiente para que empiece a moverse sin ningún problema. Y lo que es más, su velocidad no quedará entorpecida por las líneas Basic necesarias para el juego que tenga diseñado. Una vez que el sprite inicia el movimiento, su programa en BASIC o código máquina puede seguir corriendo de forma simultánea. Esto puede lograrse porque la rutina funciona a base de interrupciones, lo cual quiere decir que se ejecuta a sí misma 50 veces por segundo.

Una única línea de BASIC puede poner en funcionamiento la rutina, activándose la interrupción, con lo cual todas las líneas siguientes se ejecutarán secuencialmente, pero cada 1/50 de segundo entra en funcionamiento el programa en código máquina que controla los sprites.

Los sprites tienen la ventaja de que se pueden controlar desde el Basic sin perder efectividad.

Parámetros de los sprites

Cualquier sprite debe contener una serie de valores iniciales que posibilitan su visualización y movimiento. Estos parámetros están almacenados en forma de códigos de cartácteres en la matriz alfanumérica s$(N,M). N puede contener cualquier valor, y define la cantidad de sprites que vamos a utilizar para nuestros fines. En M se definen los parámetros propiamente dichos (8 en total).

El primer parámetro –s$(N,1)– define el número del sprite. Como ya mencionamos anteriormente, la configuración del sprite se crea de la forma acostumbrada mediante los UDGs, correspondiéndoles a cada uno de ellos 4 UDGs distintos:

Sprite 1 = UDGs A,B,C,D
Sprite 2 = UDGs E,F,G,H
Sprite 3 = UDGs I,J,K,L
Sprite 4 = UDGs M,N,O,P
Sprite 5 = UDGs Q,R,S,T

El 2.° parámetro debe inicializarse a 1. Si introducimos en él CHR$ 0, el sprite no se visualiza. Cuando se produce una colisión o se llega a uno de los bordes de la pantalla, su valor será igual al del 3.^{er} parámetro, con lo cual podremos obrar en consecuencia (hacer desaparecer el sprite, producir un rebote, sustituirlo por otro sprite, etc.).

El 3.^{er} parámetro –s$(N,3)– indica la velocidad con que nuestro sprite podrá moverse. Si contiene CHR$ 0 el sprite se inmoviliza y si queremos conseguir la máxima velocidad debemos introducir CHR$ 1.

Los parámetros 4 y 5 corresponden a las coordenadas iniciales, fila y columna respectivamente.

El 6.º parámetro incida el desplazamiento horizontal del sprite. Si queremos que la figura se desplace a razón de un caracter cada vez, CHR$ 255 indicará un desplazamiento hacia la izquierda y CHR$ 1 hacia la derecha.

El desplazamiento vertical viene definido por el valor del 7.º parámetro (255 lo hará mover un carácter hacia arriba y 1 un carácter hacia abajo).

El último parámetro es muy importante. Si su contenido es menor o igual a 64, el sprite se vuelve inactivo. Si colisiona con cualquier cosa (sea o no sprite) o llega a los límites de la pantalla, se detiene el movimiento y se desactiva; el elemento s$(N,8) contiene entonces CHR$ 0.

Empleo de la rutina

Si desea utilizar esta rutina de diseño de sprites desde el basic, una de las líneas del programa deberá ser DIM s$(N,8), siendo N el máximo número de sprites que puedan aparecer a la vez en la pantalla. Mediante un bucle for-next y unas líneas de datas, puede asignar valores a los parámetros en la matriz s$ y a continuación un RANDOMIZE USR 33013 activará la rutina, con lo cual los sprites entrarán en acción. Si en cualquier momento desea desactivarla, RANDOMIZE USR 33020 volverá al modo normal de interrupción.

Con esta rutina se puede crear 5 sprites diferentes de 16 pixels de ancho por 16 de alto.

La matriz s$ sufre una comprobación 50 veces por segundo, y mueve el sprite por la pantalla de acuerdo con los datos que contenga.

Programa demostración

Como ejemplo práctico de la utilización del diseñador de sprites, tenemos el programa de demostración en BASIC (fig. 2), el cual, una vez ejecutado, nos muestra dos bolas rebotando por la pantalla.

Pruebe a conseguir diversos efectos alterando los diferentes parámetros de ambos sprites (antes o durante la ejecución del programa) para adquirir soltura en su manejo y hacerse una idea de cómo podría emplearlos en sus propios juegos.

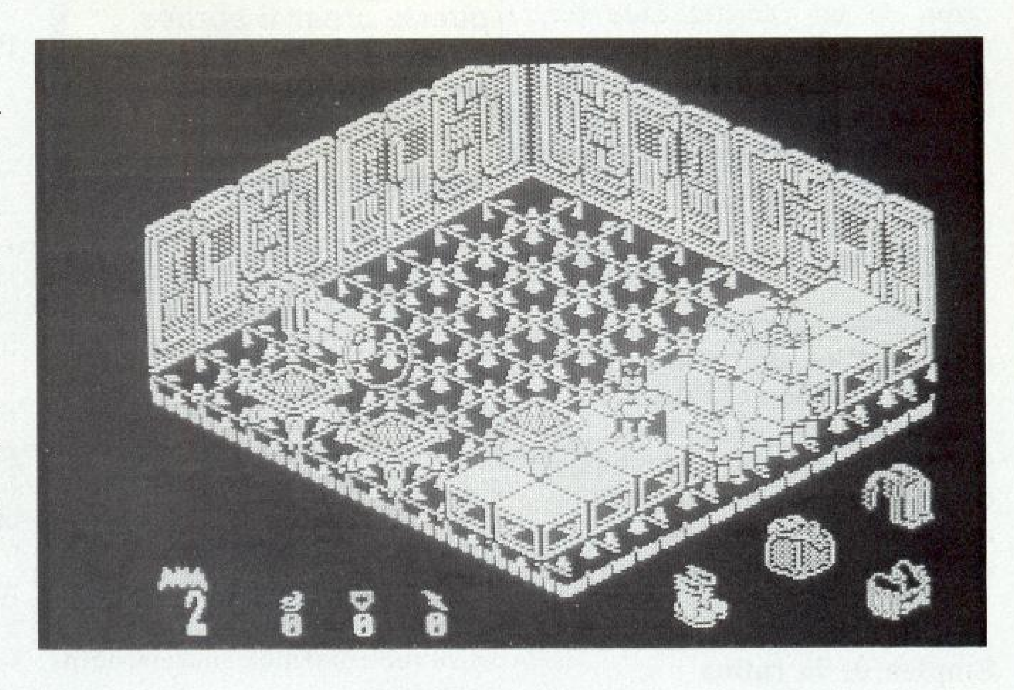
Nosotros le hemos dado la herramienta, pero ahora le toca a usted utilizarla y sacarla el máximo provecho haciendo uso de su imaginación.

Orlando Araujo Martín

BATMAN

ERBE

SPECTRUM 48K



Ocean nos sorprende muy agradablemente pasando la «barrera del Sabreman» para conseguir un juego que debe ser considerado todo un número uno de su categoría. Usando básicamente las mismas técnicas de animación, e incluso un parecido estilo gráfico, que las últimas creaciones de Ultimate, Batman demuestra que son la imaginación y la creatividad las que deben unirse a la técnica para conseguir juegos atractivos y que no aburran. Ha conseguido Ocean lo que parece que ni siquiera ha intentado Ultimate a pesar de las muchas voces que clamábamos por ello.

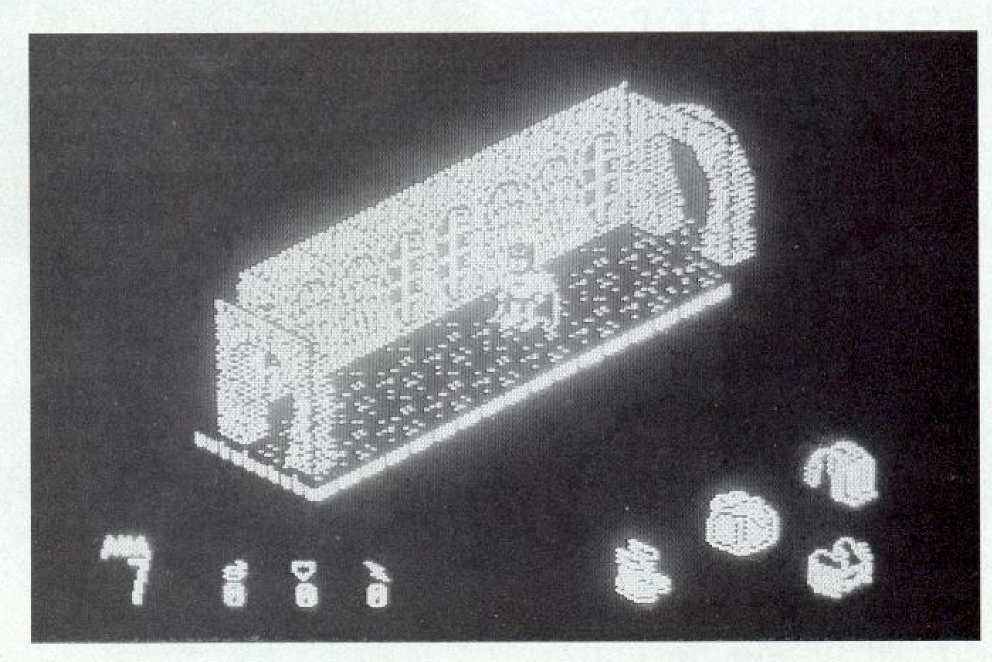
El objetivo del juego consiste, primeramente, en recoger los cuatro objetos que darán al protagonista ciertos poderes permanentes, como coger cosas, saltar, controlar la dirección de las caídas y «planear». Una vez los tengamos en nuestro poder hay que encontrar las siete piezas de la nave que permitirá a Batman salir en busca de su amigo Robin, que se haya preso por sus enemigos.

El nivel de dificultad es realmente alto, en muchas de las pantallas que componen el mapa hay que romperse la cabeza para averiguar qué secuencia de acciones y movimientos es la que nos permitirán pasarla sin que acaben con nuestras vidas. Si a esto unimos el que el mapa esté compuesto por unas 150 pantallas con dificultades de lo más variado, nos daremos cuenta de que hay que jugar muchas y muchas partidas si queremos llegar a algún sitio del que podamos sentirnos orgullosos.

Sin duda habrá ya más de uno que espere ansioso los POKEs adecuados sin los que se siente perdido.

Además de la gran calidad y variedad de los gráficos, el juego presenta un gran número de detalles que lo hacen tan completísimo como innovador. Disponemos de un interesante menú inicial con el que podemos variar algunas de las características de juego, y ciertos «Batipoderes» qaue podremos encontrar en el transcurso de la aventura y actuarán como POKEs de duración limitada.

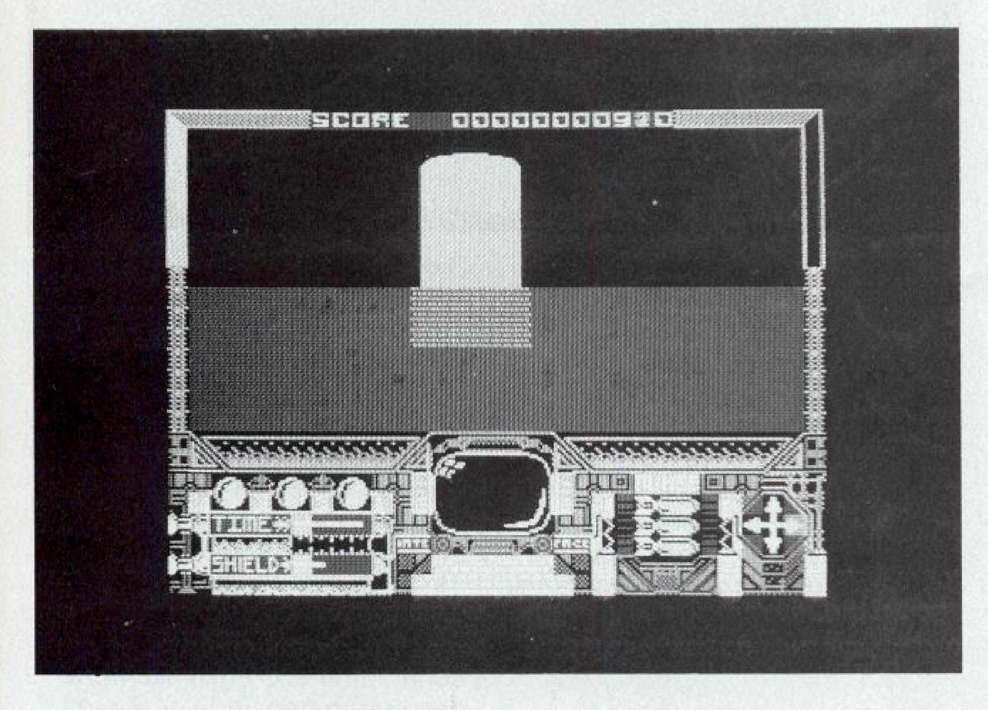
Se trata, sin duda, de un gran juego, que consigue dar a esta clase nuevos incentivos con la inclusión de un mapa más interesante, mayor variedad de movimientos y un «algo» indefinido que nos hace disfrutar como chinos en cuanto le cogemos un poco el «tranquillo».

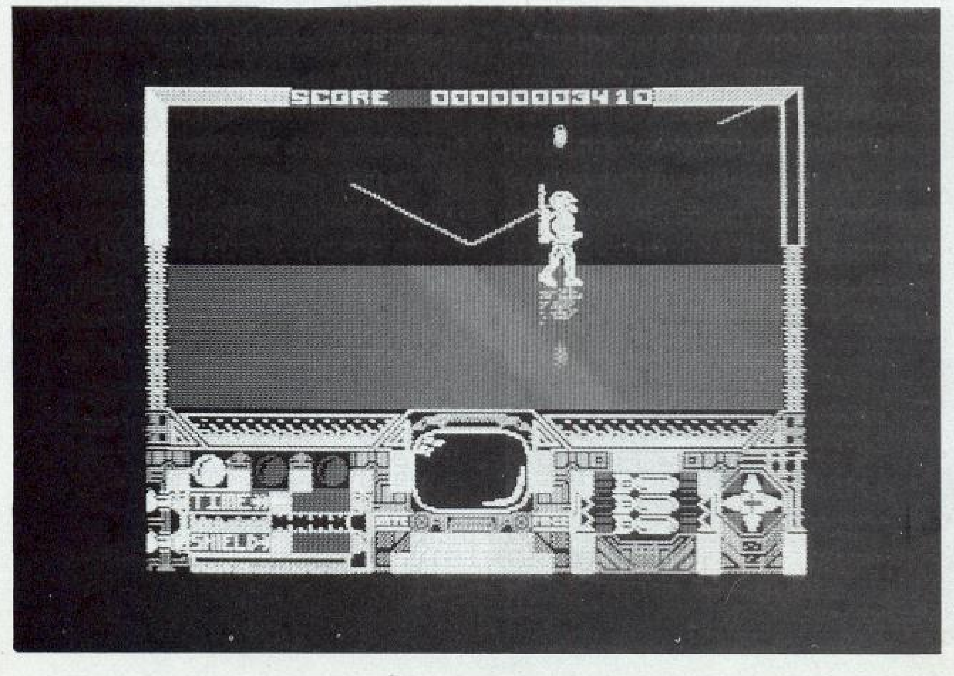


GLASS

MIND GAMES
SPECTRUM 48 K

Nos encontramos con un juego que a priori resulta interesante por salirse un poco de las reglas que parece ser dirigen un alto porcentaje de las últimas creaciones para Spectrum. Nos recuerda en algo a aquellos juegos de acción que tanto gustaban hace sólo un par de años, pero con la ventaja de que le han sido aplicadas las últimas técnicas y trucos para hacerlo más agradable de jugar.





El juego consta de varias partes con estructuras bastante diferenciadas, en las que podremos disparar nuestro laser contra naves enemigas, recorrer grandes espacios esquivando los obstáculos que nos dañen la nave y le impidan avanzar, o destruir a las salvajes ordas alienígenas que nos atacarán con peligrosas granadas hiperbáricas (¿a qué es difícil relacionar una granada con una cámara de descompresión?).

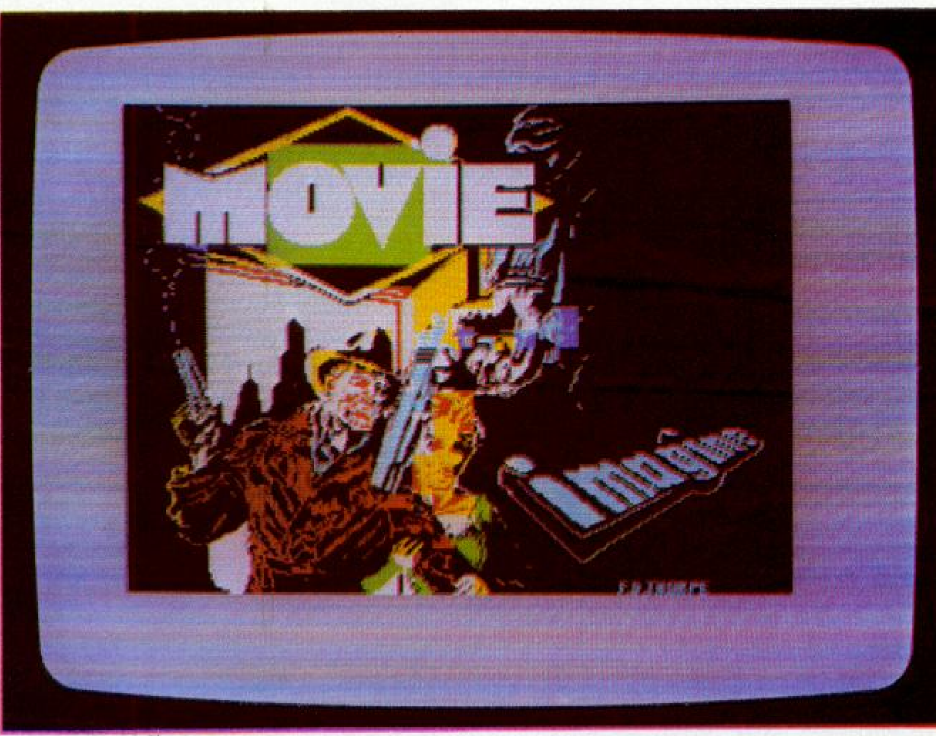
Disponemos de tres escudos que protegerán nuestra nave de cualquier tipo de impacto, pero debemos intentar usarlos lo menos posible, ya que no son eternos y sin su ayuda estamos vendidos. Lo que si parecen ser eternos son los dos electrolasers, con los que podremos combatir a los enemigos en condiciones de superioridad.

El atractivo del juego reside en la rapidez y suavidad de cada acción, y lo fácil que es al principio avanzar en la aventura.

En cuanto cubrimos unas pocas pantallas, poco a poco las cosas van poniéndose más difíciles, con lo que podemos llegar a un gran dominio de los controles en poco tiempo. Esto se une a unos gráficos en tres dimensiones sencillos pero con estilo, que hacen al juego muy agradable de llevar.

La presentación del programa es muy completa, y sobre todo el menú principal. Además, existe otro que nos permite, al acabar una partida, jugar otra comenzando en el punto en que perecimos en la anterior, con lo que comenzamos la parte correspondiente a nuestro nivel de juego sin necesidad de pasar todo el juego en cada partida.

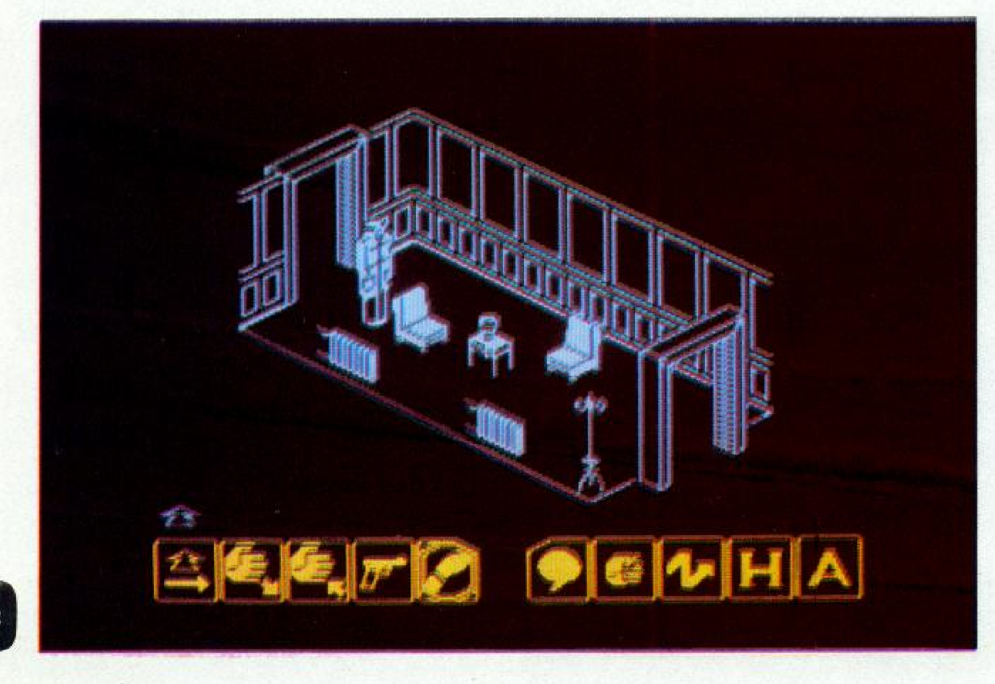
MOVIE



ERBE

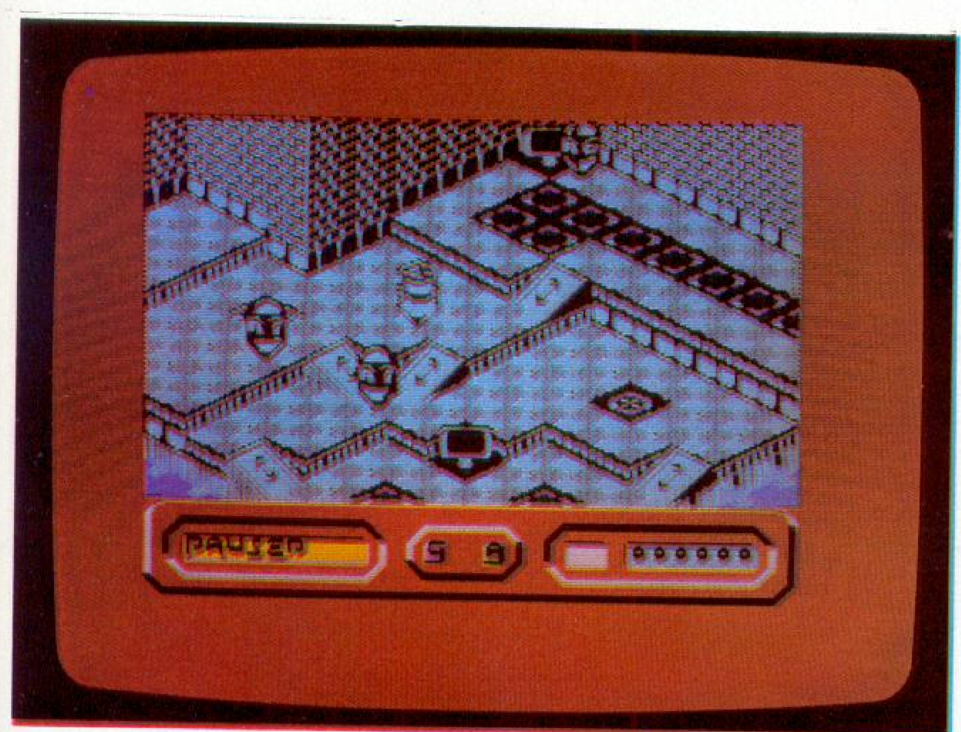
SPECTRUM 48K

Destaca este juego entre los arcade-aventuras a que nos están acostumbrando las casas de soft por los abundantes detalles innovadores que incluye. Aunque en un comienzo podamos pensar que se trata de uno más de los «ultimates» de turno, la verdad es que, cuando lo observamos más a fondo, nos damos cuenta de que, con algunos de los puntos que lo diferencian de éstos, marca una pauta que esperamos encarrile el mercado hacia nuevos modos de ver los juegos de aventura. Porque cuando comenzamos a jugar nos encontramos con que el protagonista, como viene siendo habitual, se mueve por habitaciones tridimensionales de impecable acabado en las que puede desplazar de su posición original los objetos que encuentre, etc., etc. En un principio todo parece indicar que vamos a toparnos con una aventura más, aunque ya la línea gráfica seguida puede hacernos sospechar algo: gráficos muy originales que, cuanto menos, agradecerán nuestros sufridos ojos. Pero no sólo esto llama la atención; la utilización de un menú de iconos, unida a la clásica del teclado para controlar las principales opciones del juego, da también el toque de distinción a que nos referíamos.



Sin embargo, el punto más importante que debe destacarse de este juego es, sin duda, el que podamos dialogar con los distintos personajes que encontremos como en las aventuras de corte más clásico. Y lo más curioso es que los diálogos se nos muestren por medio de «bocadillos» que salen de la boca de los personajes al estilo de comics y «tebeos». Buena técnica para hacernos sentir inmersos en la aventura que suponemos que acabarán adoptando los juegos de este tipo si quieren sobrevivir.

Como principal punto negativo, hay que reseñar que quede demasiado flojo como juego de aventura dialogada; evidentemente, el límite de la memoria de nuestro Spectrum coarta demasiado como para conseguir un juego que una un amplio mapa a buenas técnicas de animación, y que, además, incluya la suficiente Inteligencia Artificial como para conseguir diálogos «reales». Es en este sentido donde se ha tenido que «recortar» de forma que los diálogos están demasiado limitados a frases hechas y en momentos muy determinados.



QUAZA-TRON

ERBE

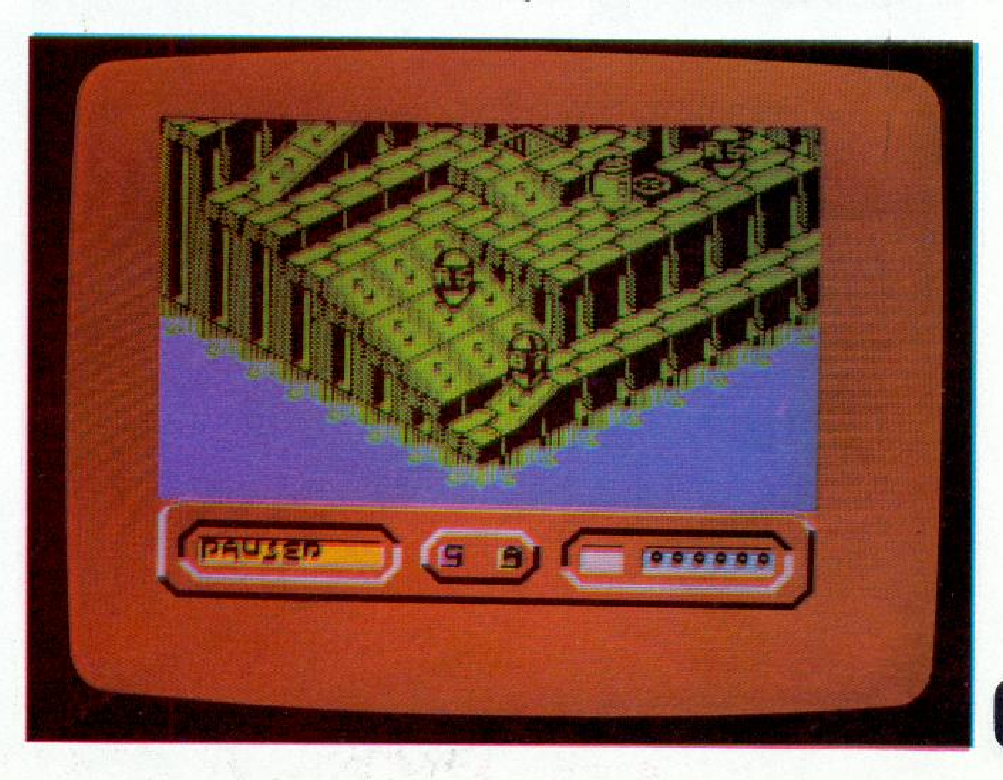
SPECTRUM 48K

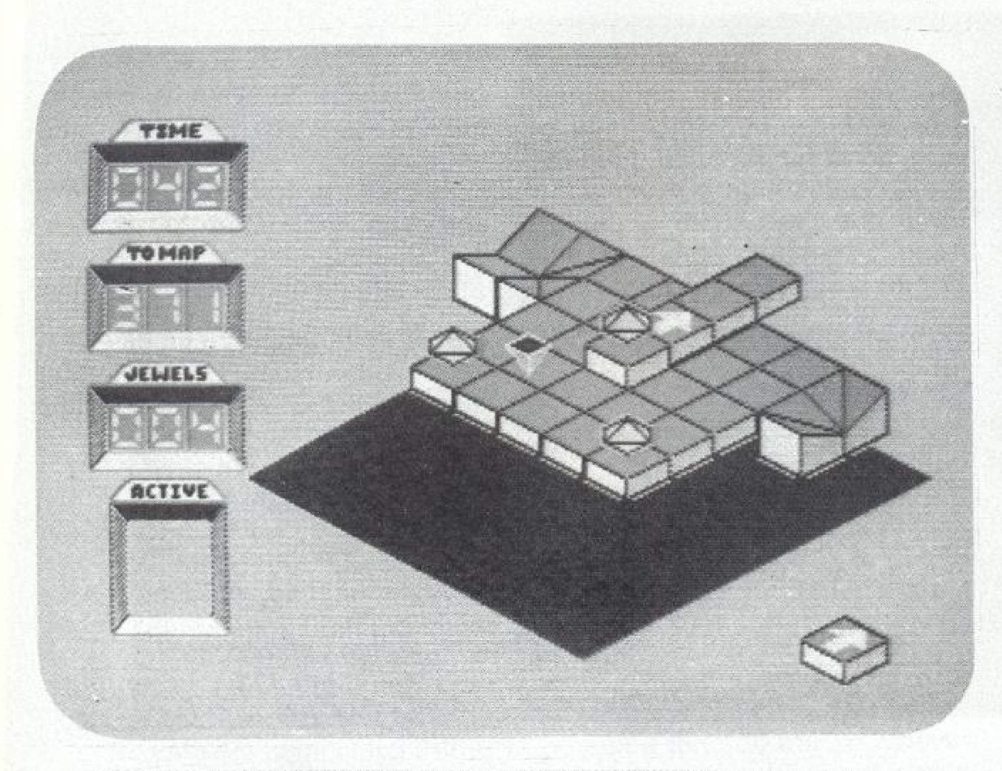
Tenemos ante nosotros a KLP-2, un droide Meknotech cuya tarea es la de desactivar los droides enemigos que viven en la ciudad subterránea de Quazatron, en el planeta Quartech. Debemos destruir al enemigo con fuego de láser, desviándolos de su ruta programada o chocando contra ellos (siempre y cuando seamos más fuertes que ellos).

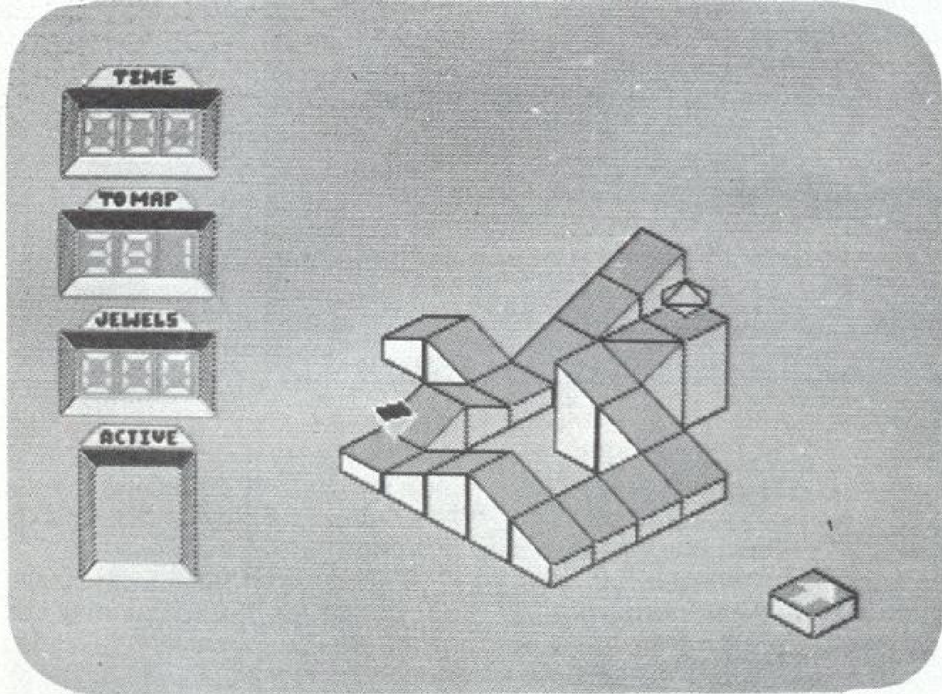
KLP-2 tiene un dispositivo experimental llamado GRAPPLE que le permite parar y destripar a los droides enemigos. Las piezas recuperadas del enemigo pueden añadirse a KLP-2 para aumentar sus recursos y duración. El sistema de ordenador enemigo puede ser penetrado para conseguir mapas de los distintos niveles, y para obtener datos sobre los droides enemigos y las piezas que los componen.

El juego se presenta bastante interesante una vez llegamos a hacernos una idea del planteamiento inicial y la forma de resolver ciertos problemas. Quizá su único defecto reseñable sea, precisamente, lo poco claras que tenemos las cosas cuando nos enfrentamos a él las primeras veces, a pesar de la agradecible traducción al castellano de las instrucciones de la carátula. Y es que no se trata en absoluto de un arcade clásico en el que baste con pasearse por los distintos niveles esquivando a los enemigos y destruyéndolos «a láser limpio»; se parece más a una arcade-aventura en este sentido, pero incluyendo detalles algo diferentes a lo que estamos acostumbrados. Aunque a primera vista pueda caerse en la tentación de compararlo con juegos como Giroscope o Spindizzy, la verdad es que son pocos los puntos en común con ellos.

El nivel técnico es suficientemente alto como para que podamos incluir esta cinta entre las últimas oleadas de grandes juegos que están invadiendo el mercado. Unos gráficos originales y que siguen una línea muy regular (aunque en ocasiones se haga algo monótona), unas secuencias de animación buenas en general, efectos sobrios pero efectivos,... evidentemente no puede decirse que sea un juego al que le falte nada importante.







SPINDIZZY

ELECTRIC DREAMS

SPECTRUM 48K

Si buscabas un juego de acción en el que no fuera necesario matar marcianos, y del que pudieras disfrutar a lo largo de docenas y docenas de pantallas con distintas y cada vez mayores dificultades, he aquí el juego que buscabas. Siguiendo la línea que inició en su día Melburne House con su magnífico Giroscope, Electric Dreams ha dado a luz un juego que supera incluso a aquél en muchos de los puntos esenciales. Un juego interesante y bien acabado que puede hacerte pasar muchas horas agradables frente a nuestro Spectrum.

Habremos de guiar al pequeño Gerald, a elegir entre una esfera, una pirámide invertida o un verdadero giróscopo, por las intrincadas rampas y plataformas de un extraño mundo artificial, recogiendo el mayor número posible de diamantes antes de que se nos eche encima el tiempo, que será nuestro principal enemigo. Cada vez que visitemos una nueva pantalla o recojamos un diamante el tiempo disponible se verá incrementado, mientras que disminuirá con las caídas y cuando hagamos uso del freno para controlar la dirección del «aparatillo».

El estilo de los gráficos en todo el juego es bastante bueno, tanto en las distintas formas que podemos encontrar cuando nos adentramos en el mapa, como en los abundantes detalles de presentación, etc. Pero lo mejor que incorpora es, sin duda, la bien conseguida sensación de realidad en los movimientos del «girador», sobre todo por la suavidad y rapidez con que los realiza y por la perfección alcanzada en la emulación de la gravedad e inercia a que se ve sometido.

Es, en definitiva, un buen juego, que debe ser recomendado a los grandes «viciosos» del joystick que disfrutan de los mapas extensos y problemáticos. El nivel de dificultad es, desde luego, muy alto si pretendemos recorrer la totalidad del mapa; se hace prácticamente imposible de acabar en condiciones normales. No obstante, siempre existe la posibilidad de armarse de paciencia, destripar el juego, y colocar los pokes adecuados para completarlo.

Indudablemente, este es uno de los mejores programas de su clase, capaz de poner a prueba los nervios y la habilidad del jugador más experimentado.

LA MAS IMPORTANTE EDITORIAL DE REVISTAS DE INFORMATICA EN CASTELLANO

El periódico INFORMATICO

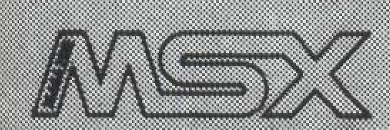
EL SEMANARIO PROFESIONAL POR EXCELENCIA

ORDENADOR POPULAR

LA REVISTA LIDER DE LOS MICROS



LA PRIMERA REVISTA EN CASTELLANO PARA IBM PC Y COMPATIBLES



LA REVISTA IMPRESCINDIBLE PARA LOS INTERESADOS EN EL STANDAR JAPONES

commodore Magazine

LA DE MAYOR DIFUSION PARA ORDENADORES COMMODORE



SINCLAIR

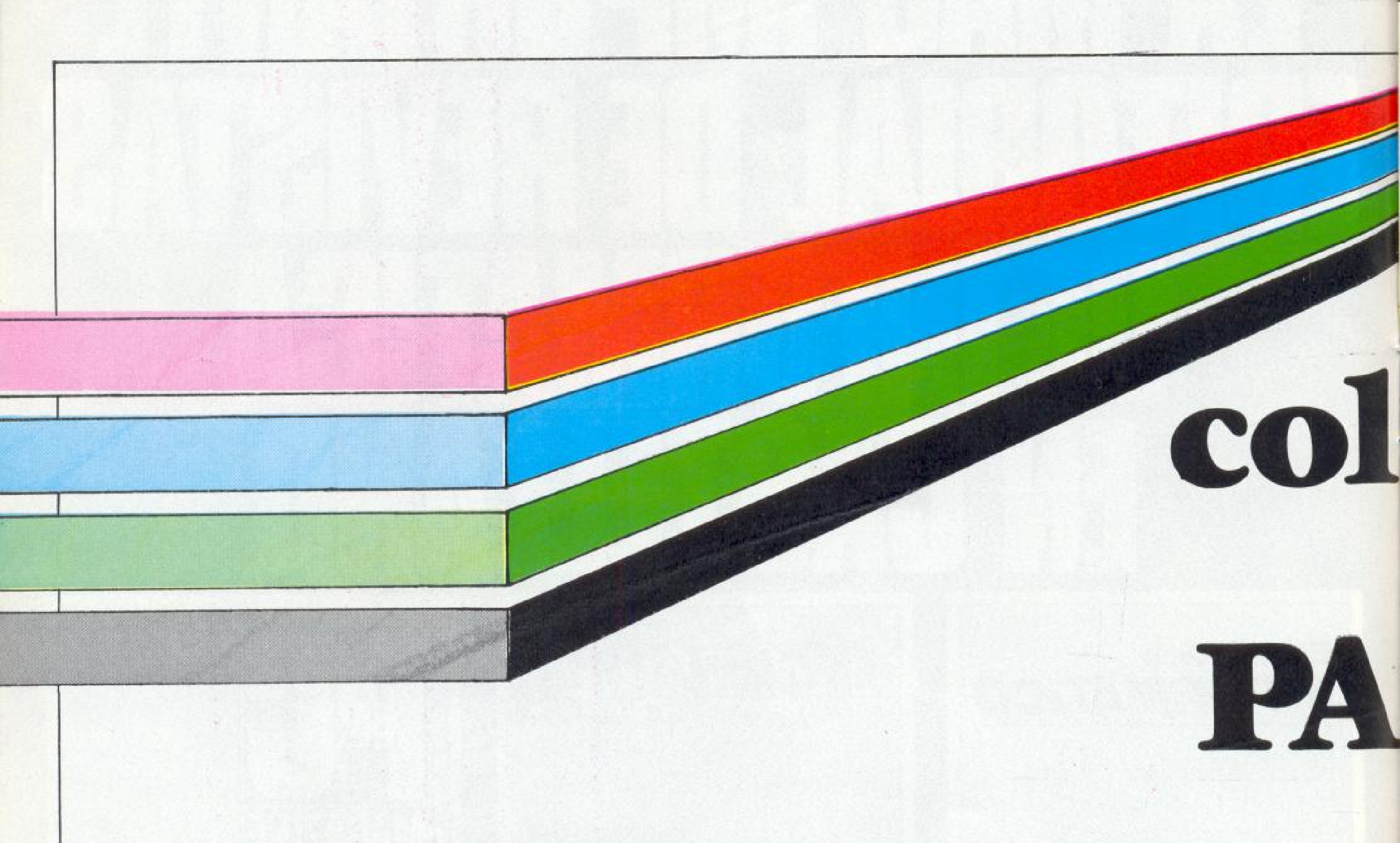
AL ALCANCE DE TODOS



EL NIVEL MAS ALTO PARA SINCLAIR

publinformática, s/a

Bravo Murillo 377 - 28020 MADRID Tel (91) 733 74 13
Pelayo 12 - 08001 BARCELONA
Tels (93) 318 02 89 (93) 301 17 00 - Ext 27 28

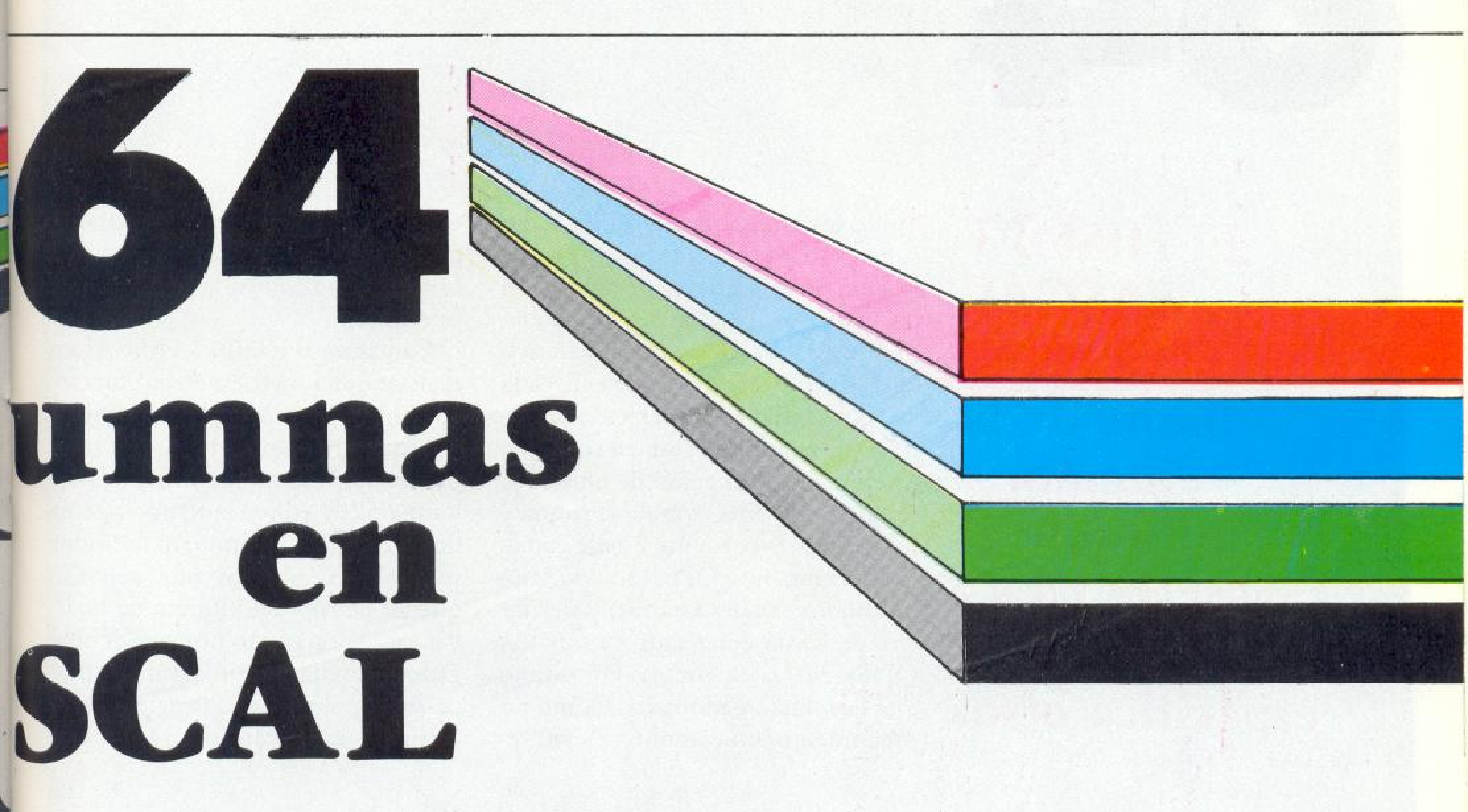


El compilador de PASCAL desarrollado por HISOFT contempla casi todas las funciones y procedimientos estándar.

Todos conocemos la extraordinaria implementación del lenguaje PASCAL hecha por HISOFT para el Spectrum. El compilador contempla casi todas las funciones y procedimientos estándar. El editor no es tan potente, pero con un poco de práctica es fácil acostumbrarse a su uso. Sin embargo, al editar surge un pequeño problema de estética. Sólo disponemos de 32 columnas para escribir el texto, y de estas las seis primeras se utilizan en el número de línea. Conforme utilizamos un poco los tabuladores para separar los distintos bloques del programa y escribimos una sentencia un poco larga, nos encontramos con que necesitamos usar una segunda línea, perdiendo el texto su aspecto estructurado que tanto facilita su comprensión.

Para solucionar esto, pensamos en la posibilidad de conseguir que todas las salidas a pantalla de este programa se hicieran a través de una rutina de 64 caracteres. La rutina ya la teníamos escrita del programa que ampliaba el sistema operativo del Spectrum. Sólo había que independizarla del resto y conseguir que el PASCAL la utilizase. Para hacerla más corta, se redujeron en parte sus prestaciones. Se eliminó la posibilidad de escribir las palabras clave del BASIC y la salida a impresora. También se cambió el puntero de posición en pantalla, que ahora es propio del programa y no lo comparte con la salida normal de 32 columnas. Por esta razón, al hacer CLS, el puntero no vuelve a la esquina superior izquierda. Por último se hizo que aprovechara las 24 líneas de la pantalla, ya que el PASCAL no utiliza las dos últimas para nada. Los códigos de control funcionan perfectamente.

Sólo quedaba por conseguir que el HISOFT PASCAL la utilice. Este pa-



ra sus salidas usa la corriente 2, que normalmente apunta al canal «S» de salida a pantalla. Vamos a explicar un poco cómo funciona esto.

En el sistema operativo las entradas y salidas se realizan a través de los «canales». Estos canales son solamente una referencia de la dirección de memoria en que se encuentra la lrutina que se va a utilizar para leer o escribir el dato. Para cada canal hay cinco bytes a partir de la dirección donde apunta CHANS en los que están codificadas las direcciones de las dos rutinas y el nombre que le identifica. Cuando encendemos el ordenador hay cuatro posibles canales que son: el «S» (Screen) que se utiliza para escritura en pantalla y si se intenta usar como entrada produce un error, el «K» (Keyboard) para leer el teclado y escribir en la parte baja de la pantalla, el «P» (Printer) para la impresora y el «R» que nos da error como entrada y como salida produce la inserción de un carácter en la posición del cursor.

Para acceder a estos canales existen 19 corrientes (16 accesibles desde el BASIC). Para cada una hay una referencia de dos bytes en el área STRMS dentro de la zona de variables BASIC. Sumando este dato a CHANS−1 obtenemos la dirección donde está la información del canal al que apunta. Con la sentencia OPEN conseguimos que una corriente determinada apunta al canal que queremos.

La corriente 2 normalmente apunta al canal «S» y es la que se usa cada vez que hacemos un PRINT sin especificar ninguna otra. También es la que usa el HISOFT PASCAL para sus salidas. Para nuestro objetivo de utilizar la rutina de 64 caracteres tendremos que conseguir que esta corriente apunte a un canal cuya dirección de

La corriente 2, que normalmente apunta al canal S, es utilizada por el HISOFT PASCAL para sus salidas.

64

El HISOFT PASCAL emplea como límite de la memoria utilizable el valor de UDG cuando no se especifica otro.

salida sea la de nuestra rutina. Aquí caben varias posibilidades. La primera sería pokear la zona de definición del canal «S», pero tiene el inconveniente de que el sistema operativo vuelve a colocar el dato correcto cada vez que realiza una operación de borrado de pantalla. También se podría insertar la información de un nuevo canal con la dirección de la rutina y luego abrir la corriente 2 a ese canal. Pero el comando OPEN sólo funciona con los canales estándar y si el interface 1 está conectado su función OPEN cuelga el sistema. Por último está la solución adoptada. Como no pretendemos utilizar ninguna impresora con el modo de 64 columnas, pokeamos la información sobre el canal «P» y abrimos la corriente 2 a este canal.

Cargando el HISOFT PASCAL en estas condiciones, en efecto funciona con 64 columnas, pero al empezar a trabajar nos encontramos con un problema. Cuando llegamos a la columna 32 el editor entiende que ha llegado al final del número de línea e inserta seis espacios que deberían quedar debajo del número de línea. Para solucionar esto nos vemos obligados a analizar el programa en busca de los pokes que lo arreglen. Están incluídos en el cargador.

```
  10 CLEAR 64461: LOAD "64col"CO
DE
  20 LET chans=PEEK 23631+256*PE
EK 23632
  30 LET c3=PEEK 23580+256*PEEK
23581
  40 POKE chans+c3,251: POKE cha
ns+c3-1,206
  50 OPEN #2,"p"
  60 RESTORE : FOR i=23296 TO 23
320: READ a: POKE i,a: NEXT i
  70 PAPER 1: BORDER 1: INK 7: C
LEAR : PRINT "Pon la copia origi
nal de HISOFT PASCAL"
  80 LOAD ""CODE 32768
  90 POKE 34665,64: POKE 34799,6
4: POKE 40525,58
 100 CLS : POKE 23675,205: POKE
23676,251
 110 RANDOMIZE USR 23296
1000 DATA 237,91,83,92,42,89,92,
43,205,229,25,33,0,128,17,22,96,
1,159,84,237,176,195,32,96
```

```
  10 DEF FN a(a$)=CODE a$-39*(a$
>"9")-48
  20 CLEAR 64461: LET n=64462: R
ESTORE
  30 FOR i=1000 TO 1210 STEP 10
  40 READ a$,a: LET s=0
  50 FOR j=1 TO LEN a$
  60 IF i<1100 THEN  LET d=16*FN
 a(a$(j))+FN a(a$(j+1)): LET j=j
+1
  70 IF i>=1100 THEN  LET d=16*F
N a(a$(j))
  80 POKE n,d: LET n=n+1: LET s=
s+d: NEXT j
  90 IF s<>a THEN  PRINT "Error
en la linea ";i: STOP
 100 NEXT i
 110 SAVE "64col"CODE 64462,1074
1000 DATA "2afefcfe20d26efcfe062
811fe082815fe092824fe0d203a2e002
4181c7dc610",3215
1010 DATA "e6f06f18142d7dfeff200
e2e3f257cfeff20062100001801 2c7df
e4038032e00",2817
1020 DATA "247cfe18300422fefcc9c
dfe0d21001718f4fe103814fe16300c1
12efcc37d0a",3354
```

64

Para utilizar la rutina de 64 caracteres tenéis que empezar por teclear el programa 1 y guardarlo en una cinta virgen con la sentencia SAVE "Pascal 64"LINE 10. Este será el cargador definitivo que se encargará de realizar todas las conexiones de los canales y cargar el HISOFT PASCAL. Antes de lanzarlo, el cargador se borra a sí mismo para dejar suficiente espacio en memoria. Gracias a esto se puede utilizar también para la carga desde microdrive. A continuación, tecleáis el programa 2. Este pokeará en memoria toda la rutina y el juego de caracteres. La tenéis que ejecutar y, si no hay ningún error, después de poco menos de un minuto estará todo dispuesto para salvarla en cinta. Hacedlo justo a continuación del cargador.

Ya está todo preparado para empezar a trabajar. Cargad los dos bloques que habéis grabado y a continuación introducir la copia original del HISOFT PASCAL. Es importante que se trate de la copia original, ya que si habéis hecho una copia de seguirdad, siguiendo las instrucciones que se dan para ello en el manual, ésta no será idéntica. Cuando se lanza el programa por primera vez, hace tres preguntas sobre la distribución de memoria (a las que normalmente no se

Para utilizar la rutina de 64 caracteres se pokea la información sobre el canal P y se abre la corriente 2 a este canal.

```
1030 DATA "11cefbc38a0afe1838043
e3f18321142fcc37d0a1148fcc3700a1
1cefbcd800a",3494
1040 DATA "2a0e5c577dfe1620036a1
8ad7c2afefc953de63f3cf53e20cd6ef
cf13d20f6c9",3896
1050 DATA "6f26002929290100fc09e
52afefc7ce618f640577c0f0f0fe6e05
f7dcb3ff583",3571
1060 DATA "5ff13e0f3801af32b2fc3
a915c06ff1f3801041f1f9f4ffdcb557
e28020eff3e",3113
1070 DATA "08e108180f3ef0a14f3e0
fb0471aa0aea91218133e0fa14f3ef0b
047afed671a",3143
1080 DATA "a0aea912afed6f0814233
d20d5ed5bfefccb0acb0acb0a7ae603c
658677ae6e0",4206
1090 DATA "cb3b836f3a8f5ce67f772
afefcc306fc0000",2274
1100 DATA "0000000002222020055OO
00005755750027471720441241102526
bf002400000",2368
1110 DATA "024444200422224000527
25000227220000002240000700000000
66001122440",1808
1120 DATA "075555700262227002512
47006161160013557100746116003465
52007122440",3024
1130 DATA "075255700255316000020
02000200224001242100007070000421
24002512020",1856
1140 DATA "06fda8700255755006565
56002544520065555600746447007474
44002547520",4192
1150 DATA "055755500722227001115
52005566550044444700577755007555
55002555520",3520
1160 DATA "065564400755577107556
65003421160072222200555557005555
52005777720",3664
1170 DATA "055225500555222007122
47007444470004623100711117002722
2200000000f",2624
1180 DATA "0254f4f00061757004465
56000344430011355300025643003464
44000355316",3280
1190 DATA "044655500206227001011
15204566550044444300057775000655
55000255520",3024
1200 DATA "006556440035553110034 4
44000342160027222100055557000555
52000577720",2816
1210 DATA "005525500055531600712
47003242230022222200621226005a00
00068bab896",3184
```

64

contesta) y después se reubica todo el bloque correspondiente al compilador y editor, con lo que los pokes serán distintos. El HISOFT PASCAL emplea como límite de la memoria utilizable el valor de UDG cuando no especificamos otra cosa. El cargador se encarga de modificar esta variable para que no sea necesario dar este dato.

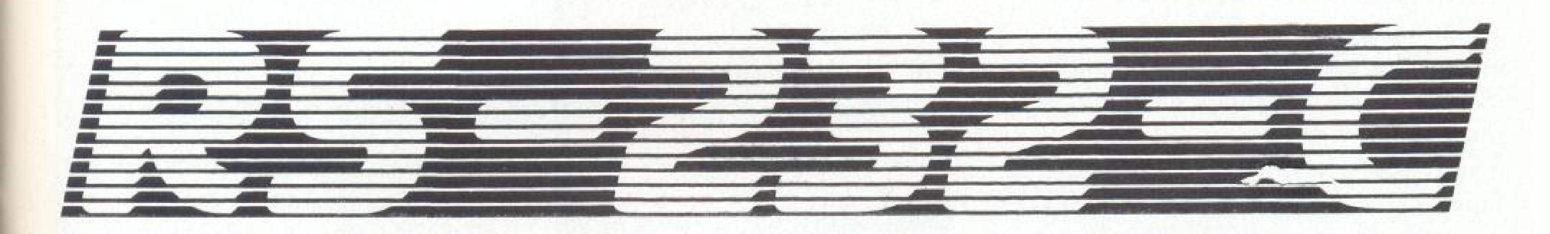
Por último queda por decir que actualmente existen dos versiones de HISOFT PASCAL. El cargador está preparado para la más moderna, que es la denominada HP4T 1.6M (es la que pone Busy... cuando está realizando alguna operación con el casete o microdrive). Para la versión anterior habría que cambiar los pokes y variar un poco el cargador, ya que necesita de un interface BASIC para funcionar. Además presenta muchos problemas para trabajar con el interface 1 conectado, aunque lo carguemos desde cinta. De todas formas puede funcionar haciendo una copia que contenga ya los pokes y utilizando un cargador que prepare la corriente 2 y luego carge el programa normalmente.

Happy Programing

Manuel Arana

La rutina de 64 caracteres trabaja únicamente en la versión más reciente del HISOFT PASCAL, la HP4TM 1.6M.

INTERFACE SERIE (I)



La denominación «RS-232-C» es ciertamente apropiada para referirse a androides galácticos de la Iniciativa de Defensa Estratégica. Sin embargo, ello no nos debe llevar a confundirlo con otros interfaces tan serios como éste, pero que transmiten datos en paralelo. Y es que la seriedad no tiene nada que ver con la serialidad.

RS-232-C

Hace ya algún tiempo —cuando los ordenadores eran de piedra—que a alguien se le ocurrió que podría aprovecharse el increíble despliegue de la red telefónica para transmitir datos entre terminales de ordeandores, que era lo que entonces más se llevaba. No mucho tiempo después —hablamos de 1967—, era posible encontrar en el mercado dispositivos que, acoplados al teléfono, realizaban la conversión de dígitos binarios a «algo» que podía transmitirse por la red telefónica con baja probabilidad de error. Esto supuso un nuevo problema: era necesario definir un estándar que permitiera la interconexión de terminales de datos y el precioso artefacto (MODEM es su nombre). En 1969, a partir de un desarrollo de la Bell Systems, la EIA publicó una serie de normas que se convertirían en el sistema actual más difundido para la transmisión de datos en serie. El RS-232-C había nacido. Las letras RS son las iniciales de «Recommended Standard», estándar recomendado. El número sirve para identificación y la letra C es un indicativo de las revisiones que ha sufrido.

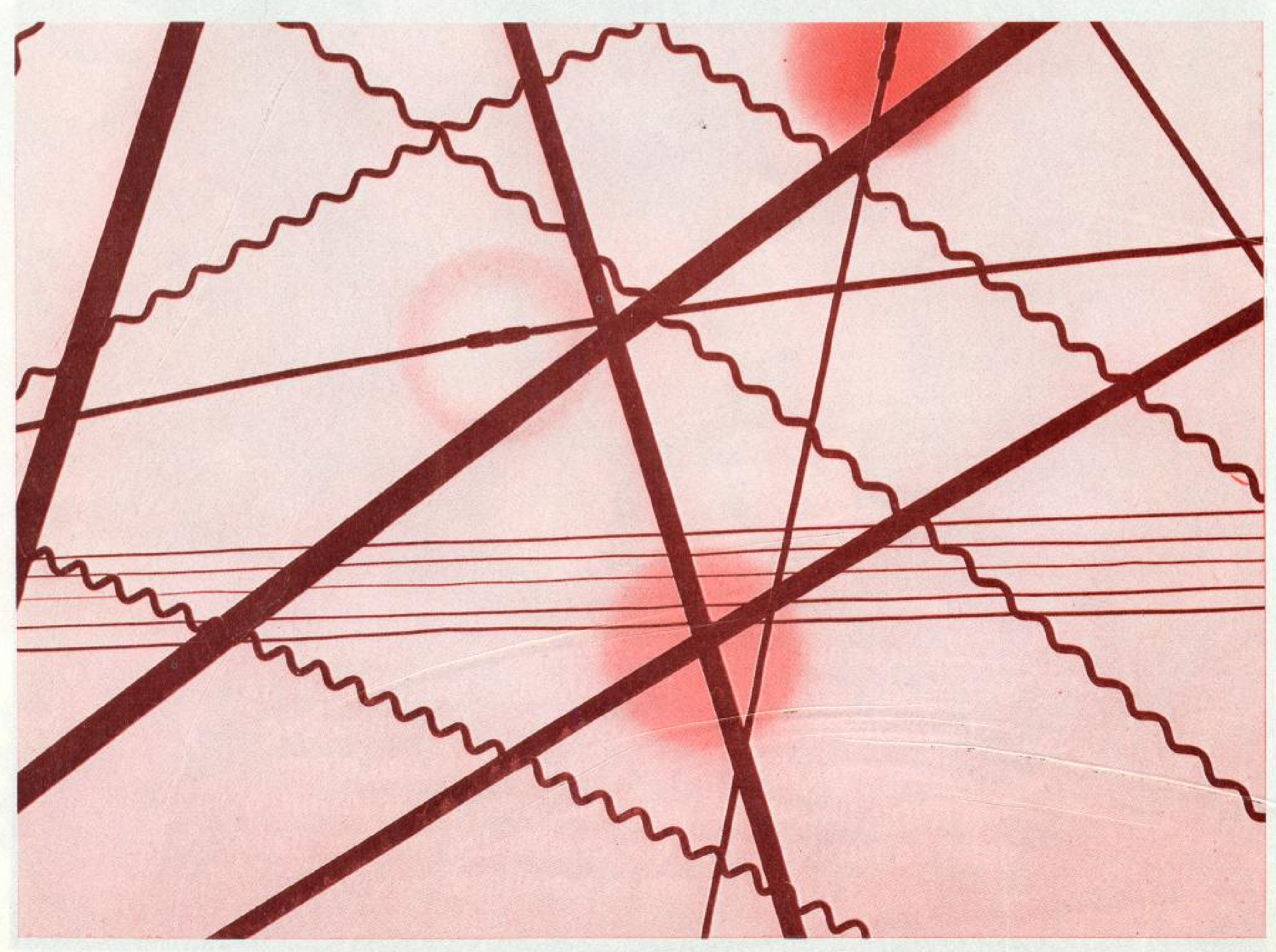
Cuando se desea transmitir datos entre dos ordenadores puede hacerse en serie o en paralelo.

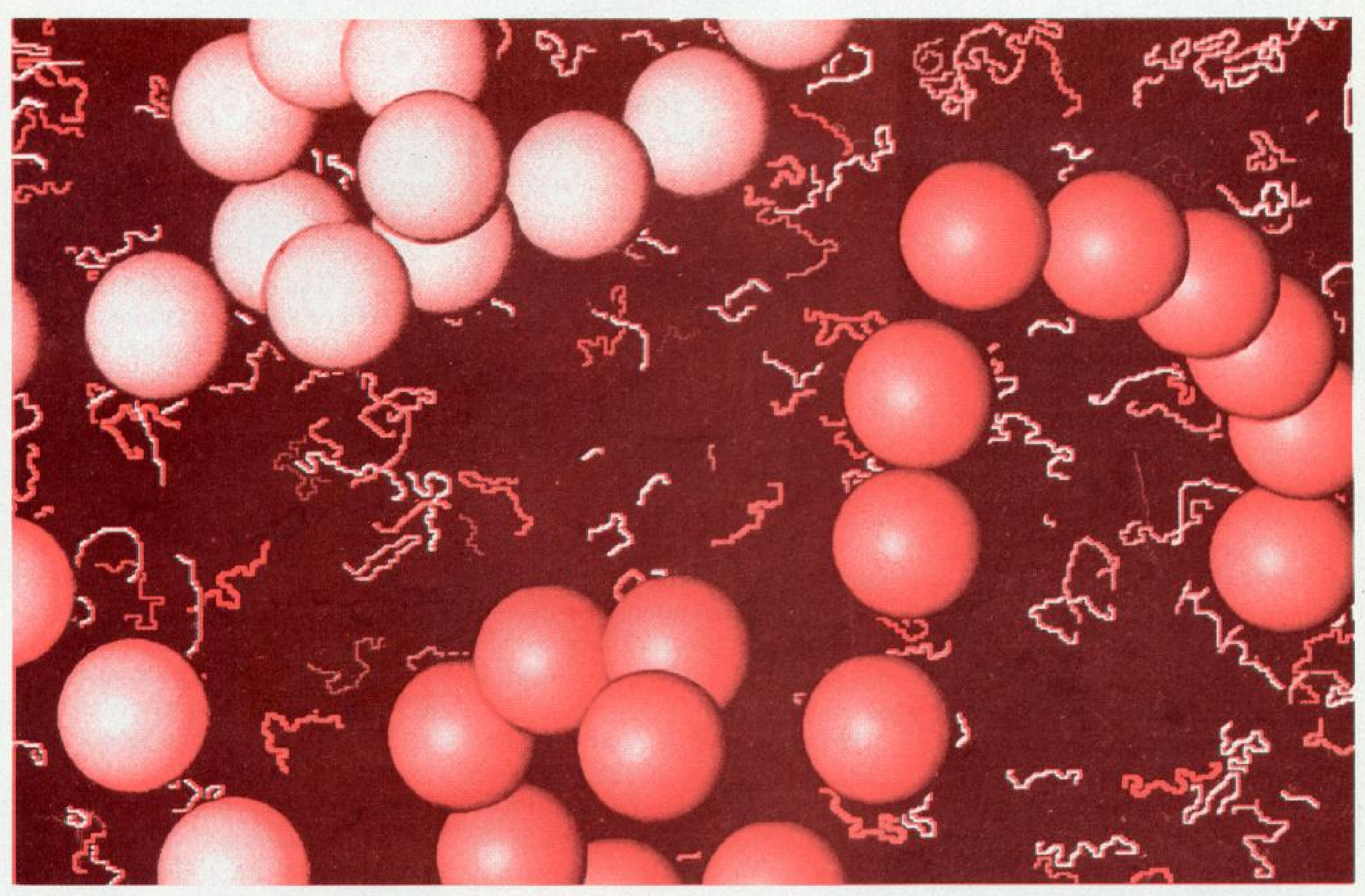
Posteriormente, el CCITT —Comité Consultivo Internacional de Teléfonos y Telégrafos— estableció normas muy semejantes bajo el nombre de «norma V-24».

Comunicación serie

Ya se ha comentado alguna vez desde estas mismas páginas en qué consisten y en qué se diferencian las comunicaciones serie y paralelo. De todos modos, creemos que no estaría de más un repaso al tema, especialmente porque resulta de extrema importancia aclarar conceptos acaso no muy asentados.

Las primeras nociones de civilización que aprenden hoy los niños al nacer es que la información digital se puede expresar como un conjunto de dígitos binarios (bits), que pueden tomar el valor 0 ó 1. Estos





unos y ceros se agrupan en palabras de modo que los ordenadores, que son unos monstruos devoradores de bits, puedan tragárselos en bloques, y no de uno en uno, lo que sería muy lento. Estas palabras empezaron siendo de 4 bits y después de 8 (tal es el caso del Spectrum), e imparablemente suben a los 16,32,... Cuando se desea transmitir datos entre dos ordenadores, podemos hacerlo enviando estas palabras de golpe (transmisión en paralelo), o bien poniendo sus bits en fila india y empujándolos por un cable único (transmisión en serie).

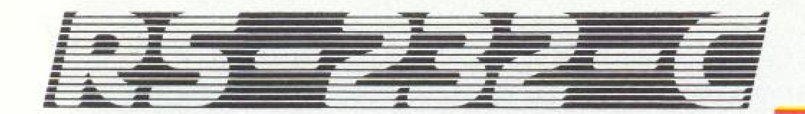
Resulta evidente que cada uno de los sistemas tiene ventajas e inconvenientes. Uno de los puntos a favor de la transmisión en paralelo es la velocidad. En igualdad de condiciones, un interface paralelo de «n» líneas será capaz de transmitir datos «n» veces más rápidamente que su homóloga serie. El inconveniente obvio que presenta, es la necesidad de contar con este número de líneas. En ciertas aplicaciones esto plantea una dificultad insalvable. Así, la transmisión a través de ocho líneas telefónicas en paralelo, no es sólo grotesca, sino que supondría problemas serios de sincronización.

La ventaja de la transmisión en serie es la reducción al mínimo del número de líneas necesarias.

Por otro lado, existen limitaciones que no son intrínsecas a la modalidad de comunicación, pero que están muy relacionadas. Tal es el caso ya mencionado del sincronismo. En general, cuando se hace un tendido con un número grande de hilos, poco importa dedicar alguno más a esta labor. Y como un sincronismo adecuado es imprescindible para la correcta recepción, la comunicación en paralelo tiene fama de ser muy fiable.

Por contra, hemos comentado que la transmisión de datos en serie ha de reducir al máximo el número de líneas —esta es su ventaja—. Por ello, la sincronización debe extraerse a partir de la propia señal recibida. Y dado que ésta se recibe distorsionada y perturbada por el ruido, no resulta en ocasiones una labor trivial. Debido a este hecho, no es de extrañar que a veces se produzcan errores. Es nuestra labor conseguir minimizarlos en lo posible.

Una vez explicada la necesidad que puede suponer la transmisión serie, es necesario definir un interface estándar que se encargue de transmitir la información binaria. La respuesta ha sido el RS-232-C.



El interface RS-232-C

Tal vez a alguien le pueda parecer extraño que estemos continuamente hablando de un **interface** cuando muchas veces ocurre que el periférico parece una parte integrante del propio ordenador. No es así, al menos si definimos este último en el sentido de «sistema procesador». El interface es simplemente de un periférico encargado de adaptar dos equipos.

A continuación debemos explicar qué significa la expresión «compatible RS-232-C». Se podía traducir como «las características eléctricas y mecánicas —conector— del interface no violan la norma RS-232-C». Esto no significa que el interface sea tan completo como el que define la EIA o el CCITT. En efecto, los estandares definen un equipo muchísimo más sofisticado lo que podría sernos útil (muchísimo significa precisamente eso). El nivel de complejidad que adoptemos depende de la aplicación que vayamos a darle.

> El RS-232 del Spectrum 128 o del Interface 1 está cableado como Data Communication Equipment.

La norma RS-232 ha sido concebida para transmitir datos entre dos equipos: el DTE: Data Terminal Equipment y DCE: Data Communication Equipment.

El interface de un extremo será siempre de un tipo y el otro del opuesto. Es absolutamente necesario identificar en una aplicación dada cuál es cada uno, por razones que veremos.

El DCE se define como el equipo encargado de establecer, mantener y finalizar la comunicación. Podemos decir que es el que tiene la iniciativa, y por tanto hace posible el diálogo —que puede ser monólogo—.

El DTE es la fuente o destino último de la información.

Somos conscientes de que estas definiciones no son lo que se dice un paradigma de claridad, por lo que vamos a poner ejemplos que disiparán nuestras dudas. Ver figura 1.

En el caso común de conectar un ordenador a una impresora, teclado o monitor (CRT, tubo de rayos catódicos)... el interface RS-232 conectado al ordenador es el DCE y el interface conectado a la impresora, teclado, CRT... es el DTE. En ocasiones se habla no del interface, sino del ordenador, la impresora,... pero ya hemos comentado que esto no es estrictamente correcto. Sin justificar la afirmación diremos que para el caso del interface propuesto, la CPU comparte labores propias del DCE. Esto se ha hecho así por razones de economía.

En el caso de comunicación entre dos ordenadores, el asunto es menos claro ya que unos modelos se

configuran como DCE y otros como DTE. El tema está en saber de qué manera ha sido realizado el nuestro. Así, el RS-232 del nuevo Spectrum-128 o del Interface 1 está cableado como DCE. Esto simplemente es el resultado de una filosofía de diseño que no tiene posteriores repercusiones ya que no hay nada que no pueda hacer un DCE que pueda un DTE y viceversa.

Más tarde comentaremos un método para descubrir cómo ha sido cableado un interface.

Si quisiéramos unir dos equipos que han sido definidos de una misma forma, esto es, dos DTEs o dos DCEs, obtendríamos lo que se ha dado en llamar un «null modem». En este caso, lo que debemos conseguir es que cada terminal «vea» su homólogo: que al DCE le parezca que está conectado a un DTE, y viceversa. Como éste sería el caso de conectar dos Spectrum entre sí, posteriormente le dedicaremos los oportunos comentarios.

El baudio es una unidad de velocidad que mide los bits transmitidos por segundo.

Tipos de comunicación

Los más importantes son:

SIMPLEX: El canal es unidireccional. Sería el caso de conectar un ordenador a una impresora.

HALF DUPLEX o SEMIDUPLEX: La comunicación es bidireccional pero nunca de modo simultáneo. El ejemplo clásico es el del «walkie-talkie».

FULL DUPLEX o DUPLEX: Los dos terminales pueden comunicarse al mismo tiempo.

Nuestro interface es capaz de establecer comunicación duplex, pero los programas sólo mantienen una modalidad semiduplex.

Velocidades

Existen un conjunto de velocidades estándar comprendidas entre 0 y 20000 baudios: 50, 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600 y 19200 baud.

El **baudio** es una unidad que mide los bits transmitidos b/8 bytes por segundo. Es un error pensar que esto equivale a transmitir b/8 bytes por segundo. Hay dos motivos por lo que esto no es así:

La transmisión se realiza en bloques de 6, 7 u 8 bits (una palabra) de forma sincrona, esto es, con un tiempo fijo entre bit y bit —dado precisamente por la velocidad de transmisión, «baud-rate»—. Por contra, la transmisión de palabras se lleva a cabo a una velocidad indeterminada, según la disponibilidad del receptor: una señal se envía cuando el equipo que la recibe ya ha digerido la anterior. Por ello se habla de una comunicación asincrona.

MODEM

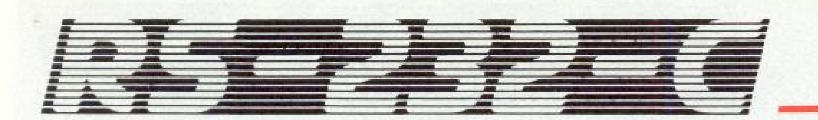
RS-232

DCE

RS-232

DTE

ORD



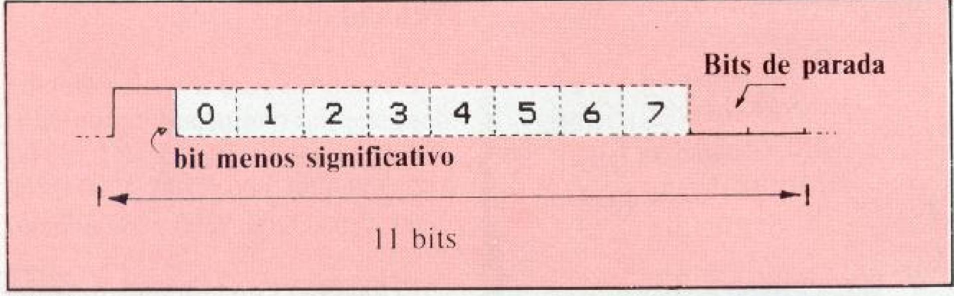


FIG. 1

El segundo es que debemos considerar que para transmitir una palabra, no sólo se envían los datos propiamente dichos. Además son necesarios bits de arranque y de parada. Para nuestro interface son 11 los bits por palabra.

Características eléctricas

Las características eléctricas se refieren a la definición de los niveles de tensión que caracterizan los estados lógicos. Son las siguientes:

estado alto/SPACE

Vout > 5 V
Vin > 3 V

estado bajo/MARK

Vout < − 5 V
Vin < − 3 V

Esto significa que, si un interface quiere significar un estado alto —Space—, tendrá que poner a su salida una tensión mayor que 5 Voltios. De manera semejante un estado bajo se hace notar por una tensión inferior a −5 Volt. Las tensiones máximas admisibles son de ±25 Volt.

Con estas definiciones se mantiene un margen de ruido mínimo de 2 Voltios, que es bastante elevado. Esto significa que si el ruido que inevitablemente está presente en el cable es inferior a 2 Voltios, aún en el peor de los casos no será posible confundir un estado por otro. Esto supone una buena protección frente a perturbaciones.

Podríamos preguntarnos el porqué de estas tensiones. Pues bien, ya hemos comentado que nos dan un buen margen de ruido mayor que si usáramos tensiones variables entre 0 y 5 Volt. que son más corrientes. Además, en la época en la que el RS-232 fue definido, era normal que estas tensiones estuvieran presentes en el ordenador. Los tiempos cambian.

Tal vez nos habrán extrañado los términos MARK y SPACE y que cada estado tenga varias denominaciones. ¡Pues,... aún hay más! El efecto más notable que produce tal profusión de nombres es el de crear la confusión total: el estado alto en realidad es el cero. Los hemos indicado porque la nomenclatura es muy corriente, pero en el futuro nos referiremos siempre a estado alto y bajo.

> En la mayoría de las aplicaciones imaginables es innecesario utilizar todas las líneas del estándar RS-232.

El nombre de MARK y SPACE, marca y espacio, viene del Morse, y son los nombres que se daban al tono y a la ausencia de tono. Y es que el código Morse fue el primer sistema de transmisión digital.

Otras características de interés

La IMPEDANCIA DE SALIDA en ausencia de tensión de alimentación será de un máximo de 300 Ohm.

La RESISTENCIA DE ENTRADA del receptor puede variar entre 3 y 7K Ohm.

La MAXIMA TENSION DE SALIDA será en cualquier caso menor que ±25 Volt.

El interface debe ser capaz de soportar un cortocircuito con cualquier otro hilo del cable sin estropearse. Nuestro interface es teóricamente capaz de resistir cortocircuitos entre dos líneas cualquiera, aunque recomendamos que no se intente comprobar. Una de las leyes de Murphy dice que no es posible construir una máquina a prueba de tontos porque... ¡son muy listos!

La MAXIMA DISTANCIA de cable aconsejada para un RS-232-C es de 16 metros. Esto viene principalmente limitado por la resistencia del cable y por la capacidad parásita que presenta. En general, a pequeñas velocidades será posible aumentar esta cifra sin problemas.

CONECTOR: la norma hace referencia a un conector tipo «D»s de 25 patillas, pero un vistazo a las listas de precios produce transtornos cardiacos. Por ello, se ha preferido usar un conector de 9 patillas del tipo usado en joysticks, compatible con el que incorpora el Interface I. De esta manera podemos usar el mismso cable que está disponible para el Spectrum.

Líneas

La discusión de las diferentes líneas con las que cuenta un interface RS-232-C es una tarea larga y tediosa. Como suponemos que nuestros lectores conocen métodos alternativos para conciliar el sueño, vamos a describir brevemente el funcionamiento de sólo aquellos que nos afectan. En este punto recordamos el comentario que hicimos en su día sobre la complejidad del interface definido como estándar. Nos referíamos fundamentalmente al número de líneas de las

que se dispone. Para el 99.9999 % de las aplicaciones imaginables, nos bastará con el propuesto.

Antes de empezar, comentaremos que el EIA y el CCITT proponen diferentes nomenclaturas. Nosotros escogeremos la del CCITT, que es la que tiene mayor difusión.

– RXD (Received Data Line). Esta línea envía datos del DCE al DTE. En ausencia de transmisión se mantiene en estado bajo.

– TXD (Trasmited Data Line). Por la línea circulan los datos transmitidos del DTE al DCE. Igualmente, está en estado bajo cuando el interface es inactivo.

– CTS (Clear To Send). Va del DCE al DTE. Se pone en alto para indicar que el DCE está listo para recibir un dato por el circuito TXD.

– DTR (Data Terminal Ready). Va del DTE al DCE. Indica a este último que el DTE está preparado para recibir datos poniéndose en estado alto.

– DSR (Data Set Ready). Indica la disponibilidad física del DCE. Se pone en alto para indicar la conexión del circuito. En general no se usa.

– GND (Ground). Es la línea de tierra, referencia para el resto de las señales.

Como se observa, la nomenclatura siempre se refiere al DTE.

Para aclarar y revisar conceptos, vamos a poner una tabla sobre el sentido de las conexiones.

	DCE	DTE
RXD	Sal	Entr
TXD	Entr	Sal
CTS	Sal	Entr
DTR	Entr	Sal

Esta tabla permite conocer de qué manera está configurado un interface dado.

El Spectrum –su interface queremos decir– cumple el sentido asignado a la columna de DCE. No resulta difícil comprender que esto

Spectr (DCE)

RXD ______ RXD
TXD ______ TXD
CTS ______ CTS
DTR ______ DTR

Impresora (DTE)

En la mayor parte de los casos no se llegará a cablear el circuito completo.

FIG. 2

Spectr 1 (DCE)

RXD ______ TXD
TXD ______ RXD
CTS ______ DTR
DTR ______ CTS

Spectr 2 (DCE)

Si quisiéramos unir dos Spectrum, tendríamos un "null modem. En este caso, tenemos que hacer que cada interface vea un DTE. Esto no plantea problemas ya que la distribución de cables es simétrica.

FIG. 3

significa que se trata de un DCE. Como las impresoras están configuradas como DTE, al menos las que conocemos, la conexión se realizará como se indica:

Descripción del Software

Formato (para Spectrum):

–no hay bit de paridad

–8 bits de datos

–bit de parada doble

Velocidad por defecto: 9600 baudios.

Vamos a explicar cómo tiene lugar el diálogo en forma genérica, visto siempre desde nuestro interfaz.

Emisión de un byte:

–Se espera a que DTR se ponga en alto. Esto significará que el DTE está preparado para recibir.

–Debemos preveer la posibilidad de que el ordenador se quede esperando una transición que acaso nunca se va a producir. Para evitarlo, debemos pensar en alguna forma de interrumpir la espera. En nuestro caso, bastará con apretar la tecla de espacio.

–Se envía un byte con el formato apropiado.

En general, cuando el DTE detecta bits de parada, pone en alto el DTR. Esto hace posible que no se reciba otro dato hasta que haya sido posible digerir el último.

Recepción de un byte

–Ponemos CTS a nivel alto. Esto indica que estamos libres.

–Esperamos a que la señal TXD se ponga en alto (bit de comienzo). Esperamos medio ciclo para leer en medio del pulso.

–Leemos los 8 bits.

–Ponemos CTS en bajo.

–Guardamos el byte.

Todos los bloques de datos que se transmitan entre dos ordenadores deberán finalizar con un CR (Carriage Return, Retorno de Carro). Su código ASCII es el 13d, 0Dh.



La ventaja de que todo el proceso esté implementado en software no es sólo económica, sino que conociendo lenguaje ensamblador, podemos adoptar los formatos, protocolos... que más nos gusten.

Ya hemos visto que conceptualmente el software es muy sencillo. En la práctica, presenta algún problema y el de el ajuste de los bucles de espera, que deben ser variables en función de la velocidad de transferencia elegida. A continuación se indica una tabla con las velocidades estándar. (Si consultamos diferentes fuentes veremos que las velocidades estándar no son tan estándar como cabría esperar).

FRECUENCIAS DE TRANSMISION ESTANDAR

VELOC (Baud)	Caract/seg
75	6.8
110	10
150	13.6
300	27.2
600	54.5
1200	109
2400	218
4800	436
9600	872
19200	1745

Es fácil darse cuenta de que las velocidades se caracterizan por ser todas el doble de la anterior. Recíprocamente, el tiempo que hay que esperar entre la transmisión de dos bits consecutivos será el doble de la correspondiente a la frecuencia mitad. Así, lo mejor será construir un bucle en el que un único recorrido produzca el retardo mínimo. Bastará con recorrerlo dos veces para conseguir exactamente el tiempo requerido... Esta ha sido la estrategia adoptada.

Con este motivo se ha construido la subrutina DELAY. Esta subrutina produce un retardo de 183*(IY+71)+138 ciclos de reloj. Teniendo en cuenta el tiempo que ocupa el programa principal, el tiempo total que se tarda en enviar un bit es 183*(IY+71)+183 ciclos de reloj. Considerando que la frecuencia de reloj es de 3.5 MHz, es fácil darse cuenta que si el contenido de IY+71 –que es una de las variables del sistema no usadas– es igual a 1, la velocidad de transmisión es de 9600 baudios. Esta es la máxima velocidad que puede conseguirse. Podríamos transmitir a 19200 bud, pero sería necesario cambiar el programa. Si es 3, la velocidad será de 4800 baudios...

Al ejecutar el programa será necesario indicar cuál es la velocidad de transmisión. Como inicialmente el puntero se pone a cero, el valor por defecto son 75 baud. Para ello se debe hacer:

```
POKE 23681,INT(19200/VT–.5)
```

(Obsérvese el redondeo)

donde VT es la velocidad de transmisión en baudios.

Para los valores estándar, el error cometido es de un 0.4%. Como el error sólo se propaga en un carácter, su efecto es indetectable.

Luis M. Brugarolas

INVERSION DE UDGs



Durante el proceso de creación de UDGs, debemos recurrir muchas veces al diseño de gráficos exactamente iguales a otros pero orientados en direcciones o sentidos contrarios. Si no se dispone de una rutina de inversión de los datos que conforman ese carácter, no nos queda más remedio que crearlos en la memoria con el consiguiente gasto de aquella. En este artículo vamos a ver paso a paso como puede crearse una rutina que realiza la inversión de gráficos de tal forma que el proceso sea comprensible para quien quiera seguirlo detenidamente.

INVERSION DE UDGs

Ante todo debemos plantearnos el problema. ¿Qué es lo que queremos conseguir? Supongamos que queremos mover una flecha en sentido horizontal. En este caso, en el momento que queramos cambiar el sentido del movimiento de la flecha (en el caso de producirse cualquier tipo de condición que contemple el programa que gobierna su desplazamiento) la punta deberá orientarse hacia el nuevo sentido del movimiento. El procedimiento más sencillo consistiría en crear dos gráficos diferentes, uno para cada sentido. Sin embargo, este método no resulta elegante y supone un gasto extra de memoria. El sentido común nos dice que debe haber alguna manera de invertir los datos del gráfico definido que queremos tratar.

Suponiendo que tuviésemos dibujada la flecha sobre una plantilla transparente bajo la forma de una cuadrícula de 8 x 8 cuadros, el nuevo UDG resultante se formaría dando la vuelta al papel de izquierda a derecha (o de derecha a izquierda, el efecto es el mismo) y esto es lo que se conoce como «efecto espejo» ya que la imagen conseguida es la misma que la reflejada en un espejo.

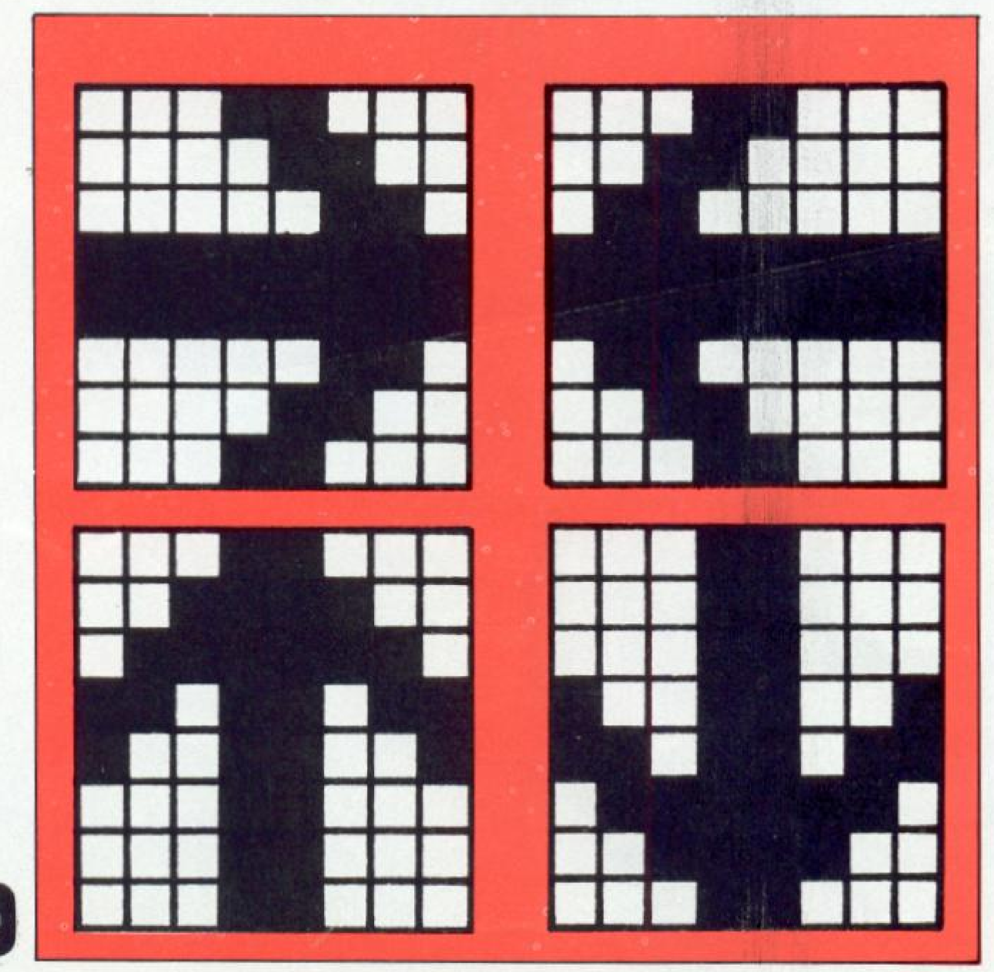
> La inversión horizontal se solucionaría sencillamente si en el juego de instrucciones del Z-80 hubiera una que produjera el «efecto espejo» en un byte.

Hasta aquí, la idea base. Si profundizamos más en el proceso, a nivel de bytes o de direcciones de memoria, descubriremos que esto es lo que ha ocurrido con cada de las ocho filas (o bytes) del UDG:

BIT 0	ocupa el lugar del	BIT 7
BIT 1	ocupa el lugar del	BIT 6
BIT 2	ocupa el lugar del	BIT 5
BIT 3	ocupa el lugar del	BIT 4
BIT 4	ocupa el lugar del	BIT 3
BIT 5	ocupa el lugar del	BIT 2
BIT 6	ocupa el lugar del	BIT 1
BIT 7	ocupa el lugar del	BIT 0

En el caso de que volviéramos a realizar el mismo proceso sobre los nuevos datos, obtendríamos por supuesto el gráfico original.

> Para crear el efecto «espejo» se debe comprobar el estado de cada uno de los bits de las ocho filas del UDG.



Cambio de lugar de las filas del UDG: inversión vertical

Otro caso de inversión supone un planteamiento y soluciones distintos del problema. Siguiendo con el ejemplo anterior, si tenemos una flecha que se desplaza en sentido vertical, la idea física de inversión consistiría en invertir la plantilla de papel, donde tenemos diseñada la flecha, de arriba a abajo (o de abajo a arriba). Al analizar más cuidadosamente la cuestión, nos daremos cuenta de que los bytes o datos nuevos del UDG así creado ya no coinciden, respecto a la colocación por filas con el anterior, tal y como sucedía en el caso de la inversión horizontal. Sin embargo, los bytes individuales no han sufrido ninguna modificación, permanecen exactamente iguales. La colocación de sus respectivos bits es idéntica:

INVERSION DE UDGs

ANTIGUO UDG		NUEVO UDG
1.ª FILA	ocupa el lugar de la	8.ª FILA
2.ª FILA	ocupa el lugar de la	7.ª FILA
3.ª FILA	ocupa el lugar de la	6.ª FILA
4.ª FILA	ocupa el lugar de la	5.ª FILA
5.ª FILA	ocupa el lugar de la	4.ª FILA
6.ª FILA	ocupa el lugar de la	3.ª FILA
7.ª FILA	ocupa el lugar de la	2.ª FILA
8.ª FILA	ocupa el lugar de la	1.ª FILA

Elaboración de la rutina de inversión horizontal

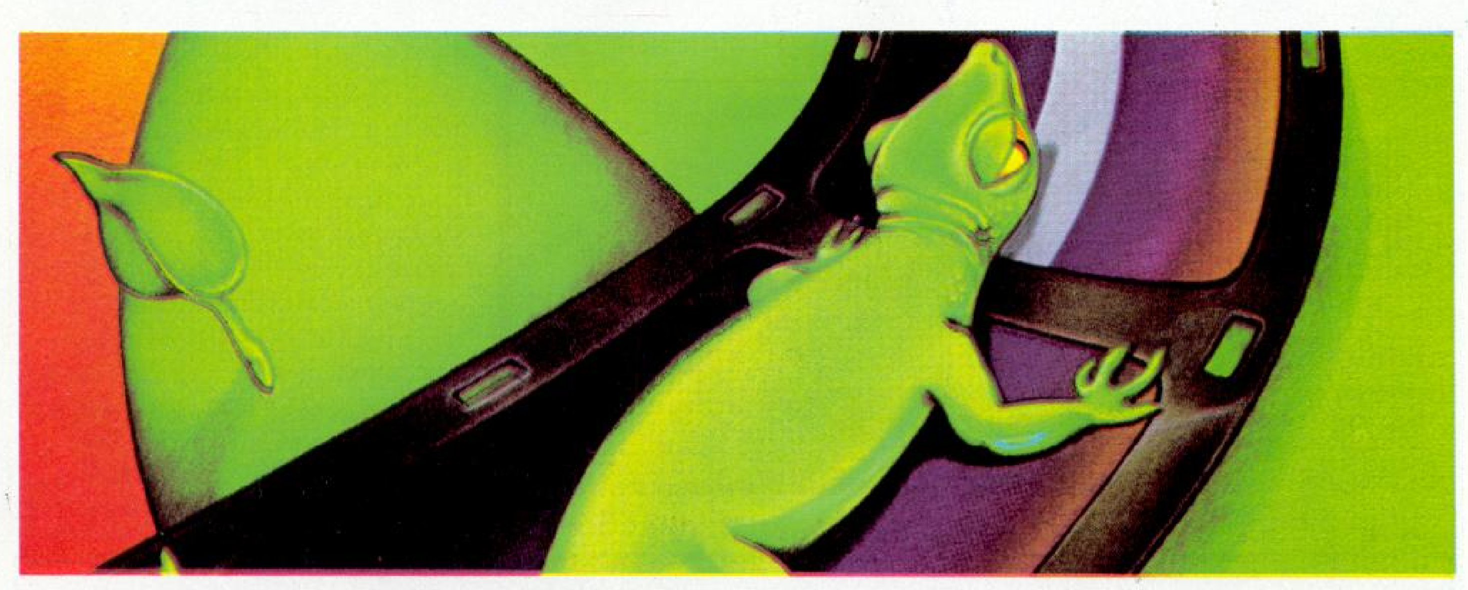
Vamos a entrar en detalles con los pormenores de la inversión horizontal de un UDG, para lo cual se deberá acudir indefectiblemente al código máquina.

Para acceder a cualquiera de las rutinas debe seleccionar el UDG que se desea invertir colocando su número de orden en una determinada posición de memoria.

El problema se solucionaría de la manera más sencilla si en el juego de instrucciones del Z-80 hubiera una que produjera el mencionado «efecto espejo» de un byte. Las instrucciones que complementan a 1 y 2 no nos sirven, ya que el resultado obtenido con ellas es totalmente diferente a lo que queremos lograr. Por tanto, gra-

```
   1 REM Orlando Araujo Martin
   5 CLEAR 59998: GO SUB 999
  10 RESTORE 12: FOR i=0 TO 15:
READ a:  POKE USR "a"+i,a: NEXT
i
  12 DATA 24,12,6,255,255,6,12,2
4
  14 DATA 24,60,126,219,153,24,2
4,24
  50 LET y=0: LET x=20: LET s=1:
 LET t=-1
 100 CLS : PRINT AT 0,6;"PULSA C
UALQUIER TECLA": POKE 59999,0: P
RINT AT 1,4; PAPER 5;"A INVERSIO
N HORIZONTAL ": RANDOMIZE USR 60
000: PRINT AT 1,27; PAPER 5;"A":
 RANDOMIZE USR 60000
 115 PRINT AT 10,y;" A "
 120 LET y=y+s
 125 IF y=0 OR y=30 THEN  LET s=
s*-1: RANDOMIZE USR 60000
 126 IF INKEY$<>"" THEN  GO TO 2
00
 130 GO TO 115
 200 CLS : PRINT AT 0,6;"PULSA C
UALQUIER TECLA": POKE 59999,1: P
RINT AT 1,4; PAPER 5;"B INVERSIO
N VERTICAL ": RANDOMIZE USR 6009
0: PRINT AT 1,25; PAPER 5;"B"
 205 RANDOMIZE USR 60090
 215 PRINT AT x,15;"B"
 217 LET x=x+t: PAUSE 1
 218 PRINT AT x-t,15;" "
 220 IF x=3 OR x=20 THEN  LET t=
t*-1: RANDOMIZE USR 60090
 225 IF INKEY$<>"" THEN  GO TO 1
00
 230 GO TO 215
 999 RESTORE 2000: LET con=0
1000 FOR i=60000 TO 60126: READ
a: POKE i,a
1001 LET con=con+a
1002 NEXT i
1003 IF con<>14015 THEN  PRINT "
ERROR EN DATAS": STOP
1005 RETURN
2000 DATA 58,95,234,203,39,203,3
9,203,39,95
2001 DATA 22,0,42,123,92,25,6,8,
126,14
2002 DATA 0,203,71,32,33,203,79,
32,33,203
2003 DATA 87,32,33,203,95,32,33,
203,103,32
2004 DATA 33,203,111,32,33,203,1
19,32,33,203
2005 DATA 127,32,33,113,35,16,21
7,201,203,249
2006 DATA 24,219,203,241,24,219,
203,233,24,219
2007 DATA 203,225,24,219,203,217
,24,219,203,209
2008 DATA 24,219,203,201,24,219,
203,193,24,219
2009 DATA 58,95,234,203,39,203,3
9,203,39,95
2010 DATA 22,0,42,123,92,25,6,8,
94,213
2011 DATA 35,16,251,175,6,8,17,8
,0,237
2012 DATA 82,209,115,35,16,251,2
01
```

INVERSION DE UDGs



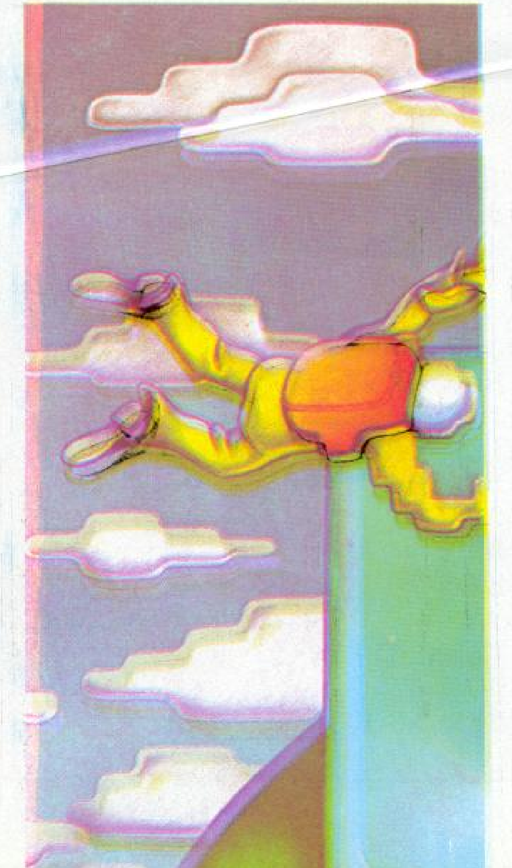
cias a las instrucciones disponibles, nos veremos obligados a comprobar el estado de cada uno de los bits de las ocho filas del UDG (64 comprobaciones), y según sea este resultado procederemos en consecuencia.

Sigamos paso a paso el listado en ensamblador (Fig. 1). Ante todo se

> La rutina de inversión vertical es bastante menos complicada y ocupa considerablemente menos memoria que la de inversión horizontal.

debe seleccionar el UDG que se desea invertir. Para ello colocaremos su número de orden en una determinada posición de memoria (59999), estando este comprendido entre 0 (para el carácter A) y 20 (para el carácter U). Una vez hecho esto (a base de un POKE desde el Basic o mediante una instrucción de carga desde código máquina) deberemos multiplicar el número almacenado por 8. Esto se consigue almacenando el número en A y realizando sobre este registro 3 desplazamientos aritméticos (SLA A) hacia la izquierda. Esta multiplicación nos servirá para acceder a la primera dirección del carácter que se haya elegido. Para ello, se carga en DE el resultado obtenido y se suma a HL, que contiene la dirección del primer dato del UDG «A» (65368). Utilizamos el registro B como contador para examinar cada una de las ocho filas del UDG. A medida que se vayan invirtiendo cada una de ellas, decrementaremos B (DKNZ) y el trabajo habrá terminado en cuanto su valor sea igual a 0.

Hasta aquí (líneas 10 a 105), no hemos hecho más que inicializar los registros principales que utiliza el programa. Lo que viene a continuación es la rutina propiamente dicha, consistente en una comprobación del estado de cada uno de los 8 bits de las 8 filas del UDG. Según el resultado de la comprobación, alzaremos o no el bit opuesto al comprobado en un registro puente (el «C» que inicialmen-



> Ambas rutinas son plenamente efectivas en el momento que tengamos que manejar muchos gráficos.

te vale 0). Una vez cargado el primer dato del UDG en «a», se procede a la primera comparación (BIT 0,A). Si este bit es igual a 0, el contador de cero del registro de estado («F») tomará el valor 0 y por lo tanto la instrucción de salto siguiente (JR NZ,SET7) no se llevará a cabo, con lo cual se procede a comparar el siguiente bit. Si por el contrario, el bit 0 es 1, el contador de cero valdrá uno y se ejecutará JR NZ,-SET7. En la etiqueta SET7 tomamos el registro C y alzamos el bit opuesto al 0, o sea, el 7.° Inmediatamente se repite hasta examinar los 7 bits y alzar los correspondientes de C si fiese necesario. Acto seguido se carga el valor de C (que contiene el dato in-

INVERSION DE UDGs

vertido) en la dirección de memoria contenida en HL (LD(HL),C), incrementándose este par de registros para acceder al próximo dato. El bucle se repite (DJNZ SIG) hasta haber completado las ocho filas, con lo cual ya hemos conseguido invertir el UDG elegido, ocupando los nuevos datos las mismas direcciones que los anteriores.

Rutina de inversión vertical

Esta rutina es bastante menos complicada y ocupa considerablemente menos memoria que la de inversión horizontal. Aquí tenemos la ventaja de que no es necesario cambiar el número de orden de los bits individuales cada byte, sino solamente cambiar las direcciones de los bytes almacenados del UDG.

En la línea 530 del listado Asembler tenemos el comienzo de la rutina que es exactamente igual al comienzo de la anterior. Se trata de acceder al UDG deseado mediante una sencilla multiplicación.

Después de cargar el registro que utilizamos como contador (el «C»), pasamos a un bucle (LOOP) que hará 8 iteraciones. Cada vez que se pasa por él, se carga el dato de la dirección del UDG –LD E,(HL)– y se almacena en el stack (PUSH DE) pasando seguidamente a incrementar HL (inc HL) para tener acceso al siguiente dato. De esta manera, hemos conseguido almacenar en el stack los 8 datos del UDG en el mismo orden de aquel (el primer dato guardado en el stack es el primero del UDG).

La siguiente operación a realizar

```
   10          ORG   60000
   20          ENT   $
   30          LD    A,(59999)
   31 ;SE CARGA EN A EL NUMERO D
EL UDG ELEGIDO
   40          SLA   A
   41 10
   50          SLA   A
   60          SLA   A
   61 ;SE MULTIPLICA POR 8
   70          LD    E,A
   80          LD    D,0
   81 ;SE CARGA EN 'DE' EL RESUL
TADO OBTENIDO EN A
   90          LD    HL,(23675)
   91 ;HL=DIRECCION DEL PRIMER U
DG
  100          ADD   HL,DE
  101 ;SUMANDO 'DE' A 'HL' SE AC
CEDE A LA DIRECION DEL UDG ELEGI
DO
  105          LD    B,8
  106 ;B=CONTADOR PARA COMPROBAR
 LAS 8 FILAS DEL UDG
  110 SIG      LD    A,(HL)
  111 ;SE CARGA EN A EL DATO COR
RESPONDIENTE DEL UDG
  120          LD    C,0
  121 ;SE LIMPIA 'C' YA QUE ESTE
 VA A ALBERGAR EL BYTE RESULTANT
E
  140          BIT   0,A
  141 ;SE COMPARA EL BIT 0 DE LA
 FILA DEL UDG CORRESPONDIENTE
  150          JR    NZ,SET7
  151 ;SI EL BIT ESTA ALZADO SAL
TA A 'SET7' Y SI ES 0 COMPRUEBA
  152 ;EL BIT SIGUIENTE DE LA FI
LA
  160 UNO      BIT   1,A
  170          JR    NZ,SET6
  180 DOS      BIT   2,A
  190          JR    NZ,SET5
  200 TRES     BIT   3,A
  210          JR    NZ,SET4
  220 CUATRO   BIT   4,A
  230          JR    NZ,SET3
  240 CINCO    BIT   5,A
  250          JR    NZ,SET2
  260 SEIS     BIT   6,A
  270          JR    NZ,SET1
  280 SIETE    BIT   7,A
  290          JR    NZ,SET0
  300 FIN      LD    (HL),C
  301 ;EL BYTE RESULTANTE SE CAR
GA EN EL MISMO LUGAR DE LA MEMOR
IA
  310          INC   HL
  311 ;SE ACCEDE AL SIGUIENTE DA
TO
  320          DJNZ  SIG
  322 ;EL SIGUIENTE DATO
  323 ;FIN DE LA RUTINA
  330          RET
  340 SET7     SET   7,C
  350          JR    UNO
  351 ;SE ALZA EN 'C' EL BIT OPU
ESTO
  360 SET6     SET   6,C
```

INVERSION DE UDGs

consiste en cargar en HL la dirección de comienzo del UDG tratado, ya que tras el bucle anterior, ésta se ha incrementado en 8. Antes de proceder a la resta, debemos asegurarnos de que el indicador de arrastre esté a 0. para ello un simple XOR A bastará, aunque cualquier otro método puede valer.

Cargamos de nuevo el contador «B» con 8 para que el siguiente bucle pueda realizar su función y por fin restamos del contenido actual de HL 8 unidades, con cual tenemos almacenada de nuevo en este registro doble la dirección original.

Tras la realización del bucle LOOP2, el UDG queda invertido verticalmente. El último dato del stack (que era el último del UDG original) se carga en la dirección donde anteriormente estaba el primer dato, y así sucesivamente.

Manejo de la rutina en sus propios programas

El programa en Basic (Fig. 2) al mismo tiempo que carga los códigos decimales en la memoria, realiza una pequeña demostración de lo que se puede lograr con esta rutina.

Cuando la quiera utilizar en sus programas deberá siempre proceder de la siguiente manera:

—Cargar en la dirección 59999 el n.º de orden del UDG a invertir (teniendo en cuenta que 0 equivale al primer carácter) con POKE,n.º de orden.

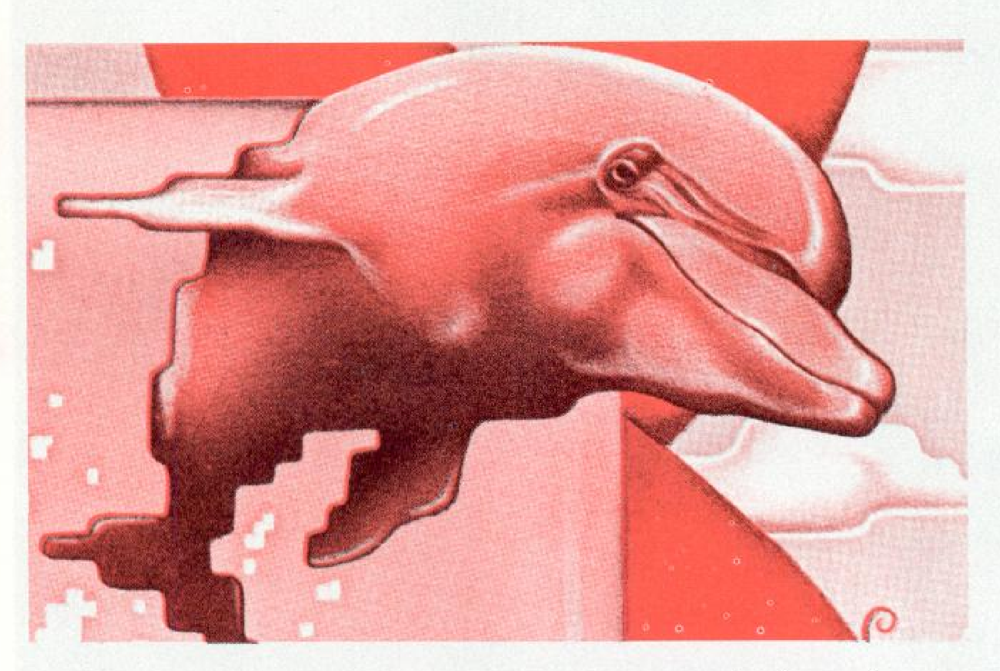
—Llamar a la rutina con RANDOMIZE USR 60000 para la inversión horizontal, o RANDOMIZE USR 60090 para la vertical.

—Imprimir el UDG transformado.

Esta rutina es plenamente efectiva en el momento que tengamos que manejar muchos gráficos, ya que si estamos empleando un único juego de UDGs (21 gráficos), su utilización no compensa el espacio ocupado por aquella (126 bytes).

```
  361 ;SALTO PARA COMPROBAR EL B
IT SIGUIENTE DE A
  370           JR    DOS
  380 SET5      SET   5,C
  390           JR    TRES
  400 SET4      SET   4,C
  410           JR    CUATRO
  420 SET3      SET   3,C
  430           JR    CINCO
  440 SET2      SET   2,C
  450           JR    SEIS
  460 SET1      SET   1,C
  470           JR    SIETE
  480 SET0      SET   0,C
  490           JR    FIN
  500
  510
  520
  530           LD    A,(59999)
  540           SLA   A
  550           SLA   A
  560           SLA   A
  570           LD    E,A
  580           LD    D,0
  590           LD    HL,(23675)
  600           ADD   HL,DE
  610           LD    B,8
  620 LOOP      LD    E,(HL)
  630 ;SE CARGA EN 'E' EL DATO C
ORRESPONDIENTE DEL UDG
  640           PUSH DE
  650 ;SE GUARDA EL DATO EN EL S
TACK
  660           INC   HL
  670 ;SE ACCEDE AL SIGUIENTE DA
TO INCREMENTANDO LA DIRECCION
  680           DJNZ LOOP
  690 ;SE REALIZA LA MISMA OPERA
CION 8 VECES (QUEDANDO EL
  700 ;UDG ALMACENADO EN ORDEN I
NVERSO EN EL STACK
  710
  720           XOR   A
  730 ;CON ESTA OPERACION SE PON
E A O EL INDICADOR DE ARRASTRE
  740           LD    B,8
  750 ;SE VUELVE A INICIALIZAR A
 8 EL CONTADOR
  760           LD    DE,8
  770           SBC   HL,DE
  780 ;RESTANDO 8 AL REGISTRO HL
VOLVEMOS A TENER ACCESO A
  790 ;LA DIRECCION INICIAL DEL
UDG
  800
  810 LOOP2  POP  DE
  820 ;RECUPERAMOS EL ULTIMO DAT
O DEL STACK
  830           LD   (HL),E
  840 ;Y LO CARGAMOS EN LA DIREC
CION DEL UDG
  850           INC  HL
  860 ;SIGUIENTE DATO
  870           DJNZ LOOP2
  880 ;REALIZA LO MISMO HASTA RE
CUPERAR LOS 8 DATOS CARGA-
  890 ;DOS EN ORDEN INVERSO
  900           RET
```

INVERSION DE UDGs



Más difícil todavía

Hasta ahora hemos visto dos maneras muy sencillas de inversión de caracteres, pero ¿y si no solamente queremos cambiar el sentido de orientación, sino también su dirección? Este planteamiento nos lleva a lo que se denomina rotación de caracteres. Una rutina que realice esta tarea hará posible la orientación de un UDG en cuatro direcciones distintas. Visualmente (llamando varias veces a la rutina e imprimiendo el carácter sucesivamente) se creará un efecto de rotación que puede llevarse a cabo tanto de derecha a izquierda como de izquierda a derecha.

El proceso es algo parecido al de la rutina de inversión horizontal, pero resulta bastante más complejo. En la segunda y última parte de este artículo veremos su realización práctica, con lo cual el lector ya estará en posesión de un conjunto de rutinas que podrá utilizar para sus propios fines y modificar según sus necesidades.

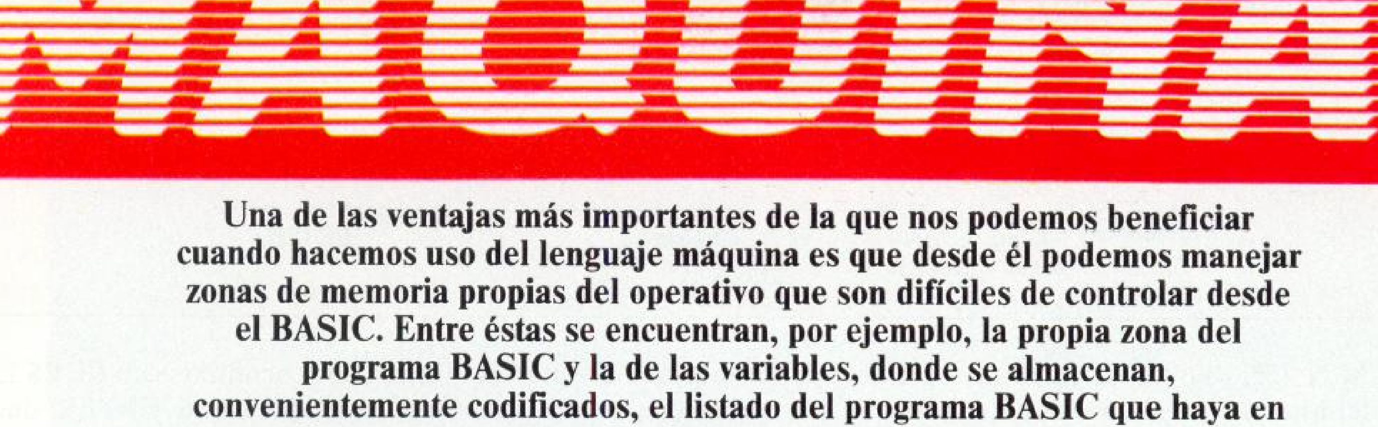
Hasta el próximo mes.

Orlando Araujo Martín



GaPw
GerbPd
GerbP
GerbS
GiantP
GibrFn
GiffHill
Gillette
GleasC
GlobM
GldNg
GldN
GldWF
Gdrich
Goodyr
GordJ
Gould
Grace
Graingr
Granitv
GtAtPc
GNIrn
GtNNk
GtWFin
GWHsp
GMP
Greyh
Grolier
GrowG
GrubEl
Grum
Guardl
Guilfd
GlfWst

APRENDIENDO

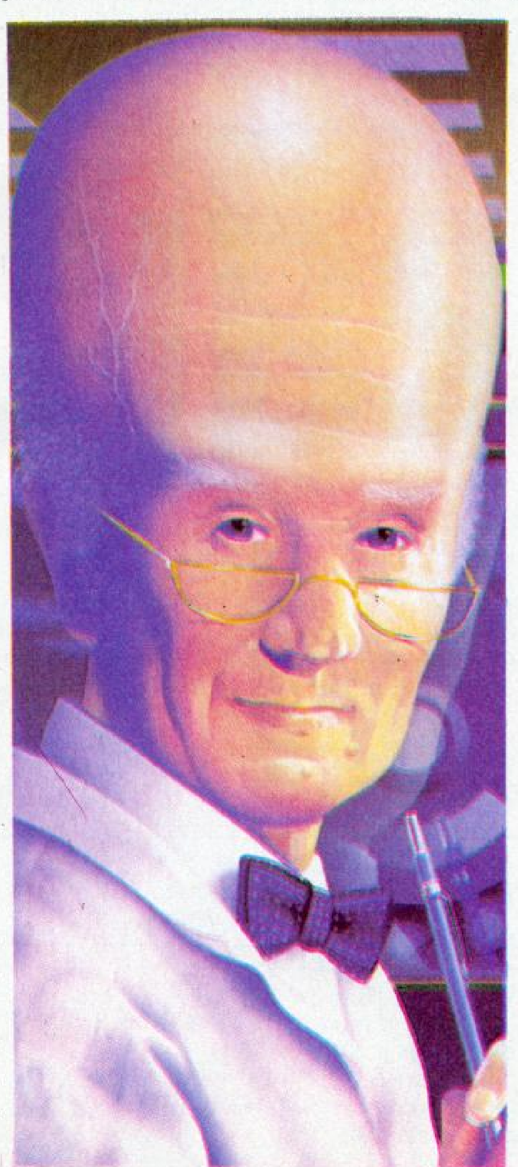


Una de las ventajas más importantes de la que nos podemos beneficiar cuando hacemos uso del lenguaje máquina es que desde él podemos manejar zonas de memoria propias del operativo que son difíciles de controlar desde el BASIC. Entre éstas se encuentran, por ejemplo, la propia zona del programa BASIC y la de las variables, donde se almacenan, convenientemente codificados, el listado del programa BASIC que haya en memoria en ese momento y los datos correspondientes a las variables que hayamos definido directamente o desde él.

Intentaremos completar en este capítulo esa panorámica que comenzamos dos números atrás sobre el mapa de memoria del Spectrum. Vamos a intentar ver, por lo tanto, todas las zonas que quedaban por encima de «Información para canales» en el diagrama que ofrecimos en el capítulo 10; zonas, en su mayoría, que son utilizadas por el operativo para conseguir que BASIC del Spectrum funcione como debe, y que pueden sernos de una gran utilidad conocer a la hora de usar las rutinas de la ROM, crear nuevos comandos, modificar los existentes o cualquier otra cosa que nos dicte nuestra imaginación.

Comenzamos pues con la zona correspondiente al «Programa BASIC», cuyo principio queda señalado por la variable del sistema PROG, en las direcciones 23635/6. Estos dos bytes nos daráan un valor que dependerá de si tenemos o no conectado el Interface 1. Si no está conectado valdrá 23755, que corresponderá al principio de la primera línea de programa si la hubiere.

El sistema usado para codificar las líneas de programa es sencillo, pero tiene algunas características especiales que es preciso destacar:

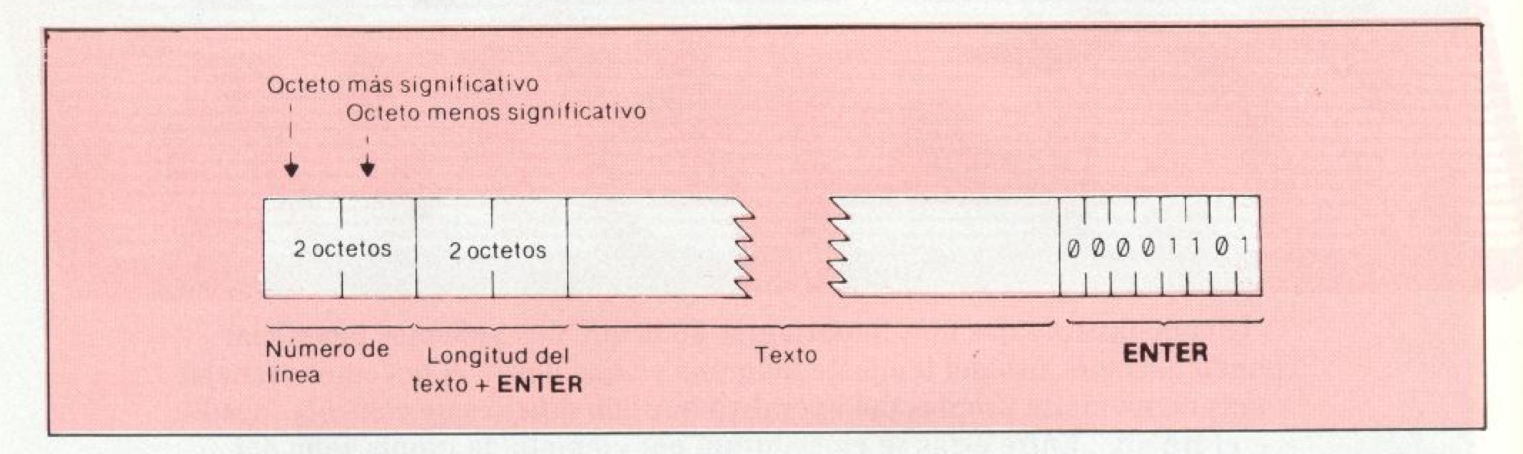


Los dos primeros bytes de cada línea corresponden al número de la misma (normalmente entre 1 y 9999), usando el clásico sistema hexadecimal para almacenarlos pero siguiendo en esta ocasión el orden contrario a lo habitual (o sea, el que sería más lógico), primeramente el byte más significativo (o de mayor peso) y a continuación el menos significativo.

Los bytes tercero y cuarto de cada línea son ocupados por su longitud en memoria desde que comienza el primer comando (quinto byte), es decir, sin contar el número de línea, incluído el "New line" (CHR$ 13, ENTER) del final. Aquí el sistema utilizado para almacenar la longitud es el normal, con el byte menos significativo en primer lugar (ver Figura 1).

A partir de aquí se almacena la línea dando a cada carácter o comando su código ASCII correspondiente tal como aparecerá en pantalla, con la única excepción de los valores numéricos, que, tras los códigos de los caracteres que los componen, incluyen un CHR$ 14 seguido de los cinco bytes correspondientes al valor en coma flotante del número que sea (ver TODOSPEC-





TRUM N.º 22, págs. 59-62 y la pág. 169 del Manual para lo de la coma flotante). Los caracteres de control (normalmente sólo los de color) no aparecerán en el listado, pero cambiarán los atributos del mismo cuando se imprima en pantalla. Los dos puntos («:») usados para separar sentencias aparece como su código correspondiente; al final de cada línea debe encontrarse un CHR$ 13 que corresponde al ENTER que pulsamos para que aquélla fuera admitida.

Número cuyo nombre es una letra única:

Bit de signo

0 1 1 | octeto para el exponente | 4 octetos para la mantisa

60ª letra | Valor

Número cuyo nombre es más largo que una letra:

1 0 1 | 0 | 1 | 5 octetos

Letra 60ª | 2º carácter | Ultimo carácter | Valor

Matriz de números:

1 0 0 | 2 octetos | 1 octetos | 2 octetos | 2 octetos | 5 octetos cada uno

60ª letra | Longitud total de los elementos y las dimensiones + 1 para el número de las dimensiones | Número de dimensiones | 1.ª dimensión | Ultima dimensión | Elementos

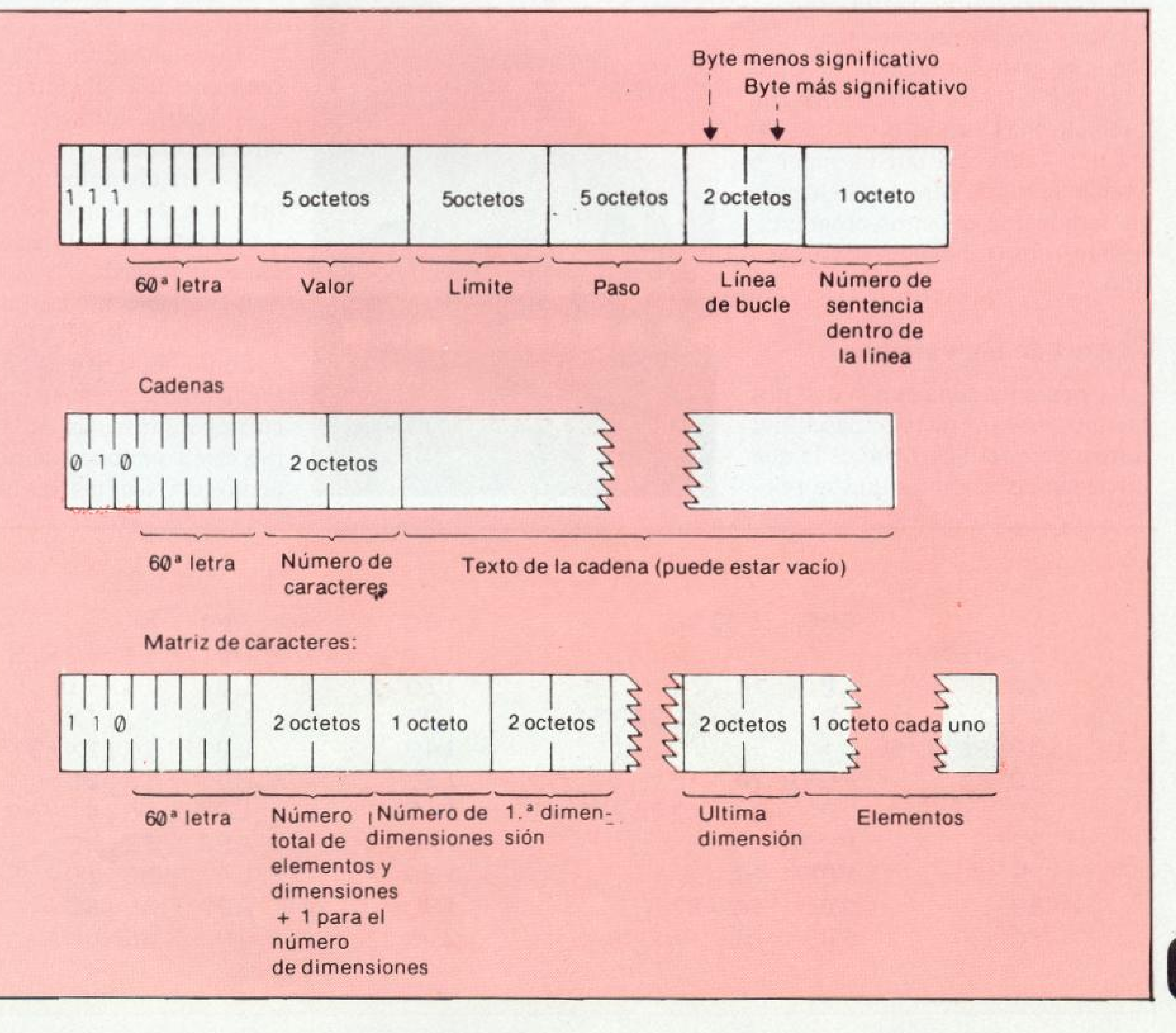
Renumerando un Programa BASIC

Hay una gran cantidad de rutinas de utilidad que pueden hacerse aprovechando los conocimientos que ya tenemos sobre la forma en que se almacenan los programas BASIC en memoria. Quizá las más clásicas por lo prácticas sean las rutinas de renumeración de líneas. Algo que se echa muy en falta en el BASIC del Spectrum, como es el comando RENUM, puede ser sustituido, como veremos ahora, por una rutina en código máquina que se encargue de esta tarea. Otros posibles usos que podemos darle a nuestros conocimientos en este sentido es, por ejemplo, crear una rutina que localice o sustituya por otra a una determinada secuencia de caracteres y/o comandos dentro del listado, variar la presentación de los valores numéricos, eliminar las líneas REM para ahorrar espacio, etc.

La corta rutina de la Figura 2 renumerará todas las líneas de un programa BASIC corriente en milésimas. El sistema que se usa es el siguiente:

En primer lugar hacemos que el par de registros HL tome el valor de línea que queramos darle a la primera de ellas; en el listado se le asigna un 10, pero es posible modificar este valor según nuestras necesidades. A continuación ponemos en el par IX la dirección de comienzo del programa BASIC tomándola de PROG (23635/6).

Seguidamente comienza un bucle que se ejecutará una vez para cada línea de programa que encontremos. En las líneas 60 a 110 lo que se hace, tras salvar HL en la pila, es comprobar si IX ha alcanzado el final del programa, lo cual lo conseguimos «comparando» IX con VARS (23627), que, al marcar el principio del área de las variables, nos indica también el final del área anterior. Es necesario efectuar aquí (antes de nada) esta operación por-



que podría ocurrir que no existiese ningún programa en memoria, lo que podría llevar, caso de no hacerse así, a que se «corrompieran» las siguientes zonas de memoria con peligro de que la máquina se bloquease.

Después de esto introducimos el nuevo número de línea (líneas 110 y 120), le sumamos a IX la longitud de la línea +4 para que apunte a la línea siguiente (líneas 130-170), sumamos a HL un número (10 en el listado) que es el incremento entre los diferentes números de línea, y volvemos a ejecutar el bucle con la siguiente línea saltando a RENUM1.

Esta rutina funcionará perfectamente con cualquier programa BASIC normal con el que la usemos, pero hay que advertir que tiene una pequeña gran limitación: no renumera los GOTOs y GOSUBs que huebiera en el listado, por lo que habrá que hacer esta tarea a mano o ampliar la rutina para que lo haga si queremos que nuestros programas «renumedos» funcionen a la perfección.

El Area de las Variables

La siguiente zona con la que nos encontramos en nuestro paseo por la memoria del Spectrum es la que almacena las variables que se utilizan desde el BASIC para guardar y manejar números o cadenas de caracteres. Conviene no confundir estas variables con las Variables del Sistema, que, como vimos hace un par de capítulos, tienen otras funciones.



El comienzo de este área viene dado por la variable del sistema VARS (23627/8), y el final por E_LINE (23641/2)–1, donde hay un byte que vale siempre 128 (80hh). La forma en que estarán codificadas las variables dependerá del tipo que sean; podemos distinguir, a estos efectos, entre seis tipos diferentes, a los que asignaremos los siguientes números de identificación:

2) Variable alfanumérica (o de cadena).

3) Variable numérica cuyo nombre tiene una sola letra.

4) Matriz numérica (numérica dimensionada).

5) Variable numérica cuyo nombre tiene dos o más letras.

6) Matriz alfanumérica (cadenas dimensionadas).

7) Variable índice (la usada en los bucles FOR-NEXT).

Como las variables en el BASIC Sinclair no distinguen entre mayúsculas y minúsculas en la letra o letras que componen su nombre, podemos emplear los tres bits más sig-

```
10          ORG   23296
20  RENUM
30          LD    HL,10
40          LD    IX,(23635)
50  RENUM1
60          PUSH  HL
70          LD    BC,(23627)
72          PUSH  IX
75          POP   HL
80          SBC   HL,BC
90          POP   HL
100         RET   NC
110         LD    (IX+0),H
120         LD    (IX+1),L
130         LD    B,(IX+3)
140         LD    C,(IX+2)
160         ADD   IX,BC
165         LD    BC,4
170         ADD   IX,BC
180         LD    BC,10
190         ADD   HL,BC
200         JR    RENUM1
```

Todospectrum



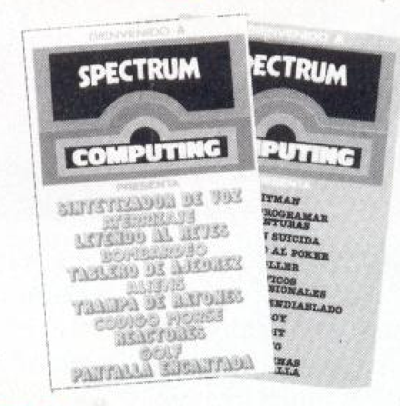
Sensacional Oferta de Suscripción

GRATIS PARA USTED SI SE SUSCRIBE A TODOSPECTRUM 2 cintas cassettes cuyo valor real es de 1750 PTAS

TODOSPECTRUM es una publicación mensual que le ayudará a obtener el máximo partido a su **SPECTRUM** y al **ZX 81**.

CONOZCA LAS VENTAJAS DE SUSCRIBIRSE A

Todospectrum



ADEMAS, le hacemos un 25 % DE DESCUENTO

sobre el precio real de suscripción (12 números)

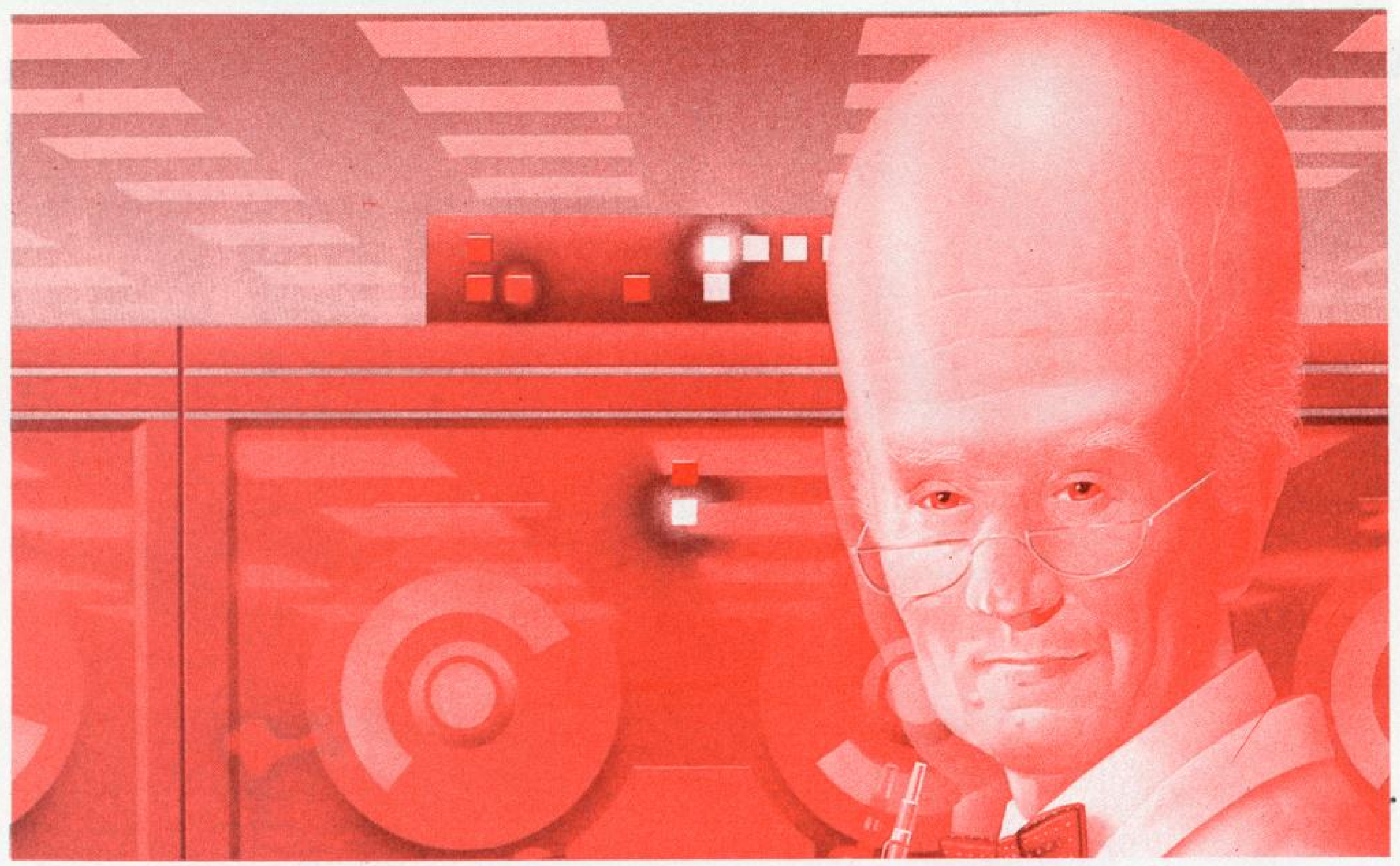
VALOR REAL DE SUSCRIPCION	OFERTA ESPECIAL DE SUSCRIPCION	USTED AHORRA
~~3.600~~ PTAS.	**2.700** PTAS.	**900** PTAS.

APROVECHE AHORA esta oportunidad irrepetible para suscribirse a **TODOSPECTRUM**. Envie **HOY MISMO** la tarjeta adjunta a la revista, que no necesita sobre ni franqueo. Deposítela en el buzón más cercano. Inmediatamente recibirá su primer ejemplar de **TODOSPECTRUM** más el **REGALO**.

Todospectrum

Bravo Murillo, 377
Tel. 733 79 69
28020 MADRID

APRENDIENDO CODIGO MAQUINA



nificativos de la primera letra de éste para especificar el tipo de variable lque es (los números de identificación en binario irían de 010 a 111), mientras usamos los cinco bits restantes para señalar de cuál de las 26 letras posibles se trata. Para conseguir el código ASCII de la letra en cuestión basta con sumar 96 al número binario formado con los bits b4-b0 del valor que tenga ésta en el área de las variables.

Una vez aclarado este punto podemos pasar a ver cómo están almacenados los números o cadenas de cada variable dependiendo del tipo a que pertenezca. Podemos empezar por las más sencillas, las variables numéricas, cuyo nombre está formado por una única letras, que ocupan un total de seis bytes (u octetos) correspondientes al tipo-nombre seguido del valor en coma flotante del número que tengan almacenado. En el caso de las variables numéricas cuyo nombre sea más largo que una letra el sistema es parecido, sólo que hay que recurrir a algo que nos permita diferenciar la última letra del nombre de las otras; el método que se utiliza es poner a cero el bit más significativo de éstas y a uno el de aquélla (ver Figura 3).

Cuando se trata de variables numéricas dimensionadas la cosa se complica; el primer byte corresponde al nombre más el tipo a que pertenece según vimos anteriormente. Los dos siguientes corresponden a la longitud total que ocupa la matriz en memoria. El cuarto indica el número de dimensiones que tiene (raramente más de dos o tres). A partir del quinto se almacenan, con dos bytes para cada uno, el valor de cada una de las dimensiones; y por último vendría el valor en coma flotante de todos los elementos ordeandos por sus subíndices de menor a mayor y de izquierda a derecha (Fig. 3).

Otro tipo cuyo formato en memoria es inevitablemente complicado es el compuesto por las variables índice, ya que deben almacenar varios valores además del que tengan en cada momento. Tras un primer byte formado por el tipo-nombre, aparece el valor de la variable seguido del límite que se dio con el comando FOR cuando fue definida y el paso (STEP) que se ha de sumar en cada NEXT (uno en la omisión), como valores en coma flotante, además del número de línea (dos bytes) y el de sentencia dentro de la línea (un byte) a donde se debe saltar para cerrar el bucle (sentencia que sigue al FOR en que se definió).

Las variables de cadena son más sencillas: un primer byte con el tipo y el nombre, dos bytes más que indican la longitud de la cadena, y los códigos ASCII correspondientes a los caracteres de la misma. Cuando están dimensionadas siguen un for-

mato similar a las matrices numéricas, considerándose a cada carácter de una cadena como un elemento (ver Fig. 3).

Otras Zonas de Memoria

Entre las siguientes zonas de memoria que podemos encontrar si seguimos «subiendo» hacia las zonas altas hay algunas que usa el operativo de forma bastante dinámica. Entre ellas podemos citar a la que queda entre E_LINE y WORKSP, que es la zona donde se almacena lo que escribimos cuando tecleamos una orden directa o editamos una línea de programa. El formato usado es el más sencillo posible: los códigos que corresponden a cada carácter o palabra-clave seguidos de un 128 (80h) para finalizar. Si la orden ha sido introducida (con ENTER) el carácter de éste (13) aparecerá también.

Tras esto nos encontramos con el espacio de trabajo y de entrada de datos, que es el lugar a donde apunta la corriente R, usada desde el código máquina para imprimir en este área como vimos en el capítulo pasado. Esta zona se usa, entre otras cosas, en los INPUTs, para almacenar el nombre de un programa que está siendo cargado de cinta, etc.

Sobre esto, y tras la pila del calculador (que será tratada en otra ocasión), está el denominado espacio de reserva, entre STKEND y la pila de máquina, que es el «hueco» que permite crecer al programa BASIC «empujando» a las otras zonas hacia aquí. Este espacio lo podemos limitar modificando RAMTOP mediante el comando CLEAR, de forma que reservemos del avance del BASIC una determinada cantidad de espacio para nuestros programas en máquina. Obsérvese cómo existe sobre el stack o pila de máquina otra pila llamada «de GOSUB»; aquí es donde se almacena los números de línea y sentencia de cada GOSUB que sehaga, con el fin de volver al sitio desde donde se hizo la llamada cada vez que se ejecute un RETURN. Tanto ésta como el stack crecen hacia abajo en el espacio de reserva sin límite ninguno. Es por esto es que sea aconsejable siempre calibrar que el número de RETURNs que se ejecuten en un programa sea el mismo que el de GOSUBs, y en código máquina, que haya tantos RETs y POPs como CALLs y PUSHs, ya que sino estas pilas podrían crecer tanto hacia abajo que invadieran zonas de memoria que no les corresponden; además de que está el riesgo de que un RET lleve el control de un programa a zonas imprevisibles de la memoria, con la consecuencia más que probable de que se produzca un incómodo bloqueo del sistema.

Luis Gala

SIETE Y MEDIA

**Usted deberá ganar al ordenador, que siempre juega de banca, mediante nueve apuestas como máximo.
Tiene tres opciones:
a) Jugar una carta vista. b) Jugar una carta oculta. c) Plantarse.
Toda carta vista que usted juegue, el ordenador conocera su valor.
Toda carta oculta que usted juegue, el ordenador ignorará su valor.
Al salir la primera carta de cada apuesta, usted dirá la cantidad de duros que juega, hasta un máximo de 18.**

Francisco Moreno

```
   0>REMPACO MORENO MONTEVERDE
  20 POKE 23658,8: POKE 23609,10
0
 110 GO SUB 8860
 120 GO SUB 1600
 130 GO SUB 2000
 140 GO SUB 1000
 150 GO SUB 1400
 170 GO SUB 3000
 180 GO TO 1500
 990 STOP
1000 REM Presentacion Pantalla
1010 BORDER 5: PAPER 4: INK 0: C
LS
1020 DRAW 255,0: DRAW 0,175: DRA
W -255,0: DRAW 0,-175
1030 FOR a=28 TO 240 STEP 24: PL
OT a,0: DRAW 0,175: NEXT a
1400 REM Graficos
1410 RESTORE
1420 FOR a=0 TO 7: READ n: POKE
USR "A"+a,n: NEXT a
1430 FOR a=0 TO 7: READ n: POKE
USR "B"+a,n: NEXT a
1440 FOR a=0 TO 7: READ n: POKE
USR "C"+a,n: NEXT a
1450 FOR a=0 TO 7: READ n: POKE
USR "D"+a,n: NEXT a
1460 DATA 0,34,119,127,62,28,8,0
,0,16,56,124,124,16,16,0,0,8,28,
62,62,28,8,0,0,28,28,127,127,8,8
,0
1470 RETURN
1500 REM Otro juego
1505 BORDER 5: PAPER 4: INK 0: C
LS
1510 PRINT AT 10,10;"JUEGA OTRA
VEZ? S/N"
1520 PAUSE 0: CLS
1530 IF INKEY$="S" THEN  GO TO 1
```

```
30
1540 STOP
1600 REM Instrucciones.
1605 CLS
1610 PRINT                        "
   Usted debera ganar al ordena-
dor, que en este caso siempre
juega de banca, mediante nueve
apuestas como maximo."
1615 PRINT
1620 PRINT                        "
   Tiene tres opciones:
a) Jugar una carta vista
b) Jugar una carta oculta
c) Plantarse"
1630 PRINT
1640 PRINT                        "
   Toda carta vista que usted
juegue, el ordenador conocera su
valor."
1650 PRINT
1660 PRINT                        "
   Toda carta oculta que usted
juegue, el ordenador ignorara su
valor."
1670 PRINT
1680 PRINT                        "
   Al salir la primera carta de
cada apuesta, usted dira la can-
tidad de duros que juega, hasta
un maximo de 18."
1690 PRINT #1;AT 0,0;"PULSE UNA
TECLA PARA CONTINUAR": PAUSE 0:
INPUT O
1700 CLS
1710 PRINT
1720 PRINT                        "
   El ordenador juega en la pri-
mera columna de la izquierda,
una vez que usted a finalizado
su juego, apareciendo en su pan-
talla el resultado de sus apues-
tas."
1730 PRINT
1735 PRINT                        "
   Si la puntuacion del ordena-
dor supera a las siete y media o
es inferior a las siete y media,
las apuestas que usted tenga con
un valor igual a las siete y me-
dia se pagaran con el doble de
su apuesta."
1740 PRINT '''                    "
       !! BUENA SUERTE !!"
1750 PRINT ''                     "
PULSE UNA TECLA PARA CONTINUAR"
1760 PAUSE 0
1770 RETURN
1900 STOP
2000 REM Nombre del jugador
2005 BORDER 5: PAPER 4: INK 0: C
LS
2010 INPUT "INTRODUZCA SU NOMBRE
";n$
2020 IF LEN n$>7 THEN  PRINT AT
20,0;"INTRODUZCA MENOS DE 8 LETR
AS": PAUSE 300: CLS : GO TO 2010
2030 RETURN
3000 REM Juego del jugador
3001 LET xx=0: LET cr=0: LET oc=
0: LET q=7: LET cartas=0
3002 LET l=-1: LET c=4: LET p=0:
 LET g=0: LET z=0: LET p1=0: LET
 p2=0: LET p3=0: LET p4=0: LET p
5=0: LET p6=0: LET p7=0: LET p8=
0: LET p9=0
3003 LET w1=0: LET w2=0: LET w3=
0: LET w4=0: LET w5=0: LET w6=0:
 LET w7=0: LET w8=0: LET w9=0: L
ET numero=0: LET ap=0: LET r=0
3004 LET ap1=0: LET ap2=0: LET a
p3=0: LET ap4=0: LET ap5=0: LET
ap6=0: LET ap7=0: LET ap8=0: LET
 ap9=0
3010 IF numero=4 THEN  LET l=l-2
: LET cartas=cartas-1: LET p=0:
GO TO 3021
3012 INPUT "N. DE APUESTAS(MAX.9
)";g
3014 IF g>9 THEN  GO TO 3012
3016 IF z=1 THEN  GO TO 3020
3020 IF r>=1 THEN  GO TO 4200
```

```
3021 PRINT #1;AT 0,0;"VISTA, OCU
LTA O SE PLANTA(V/O/P)": PAUSE 0
: INPUT O
3022 IF INKEY$="V" THEN  GO TO 3
029
3023 IF INKEY$="O" THEN  LET r=1
: LET q=1: LET oc=1: LET xx=1
3027 IF INKEY$="P" THEN  LET oc=
0: GO TO 4000
3029 FOR x=-10 TO 0: BEEP .0125,
x: NEXT x
3031 FOR y=0 TO -5 STEP -1: BEEP
```

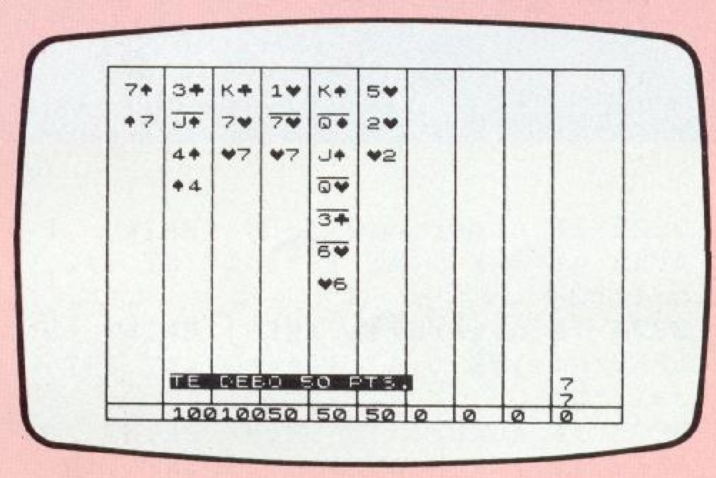
Usted debera ganar al ordenador, que en este caso siempre juega de banca, mediante nueve apuestas como maximo.

Tiene tres opciones:
a) Jugar una carta vista
b) Jugar una carta oculta
c) Plantarse

Toda carta vista que usted juegue, el ordenador conocera su valor.

Toda carta oculta que usted juegue, el ordenador ignorara su valor.

Al salir la primera carta de cada apuesta, usted dira la cantidad de duros que juega, hasta un maximo de 18.



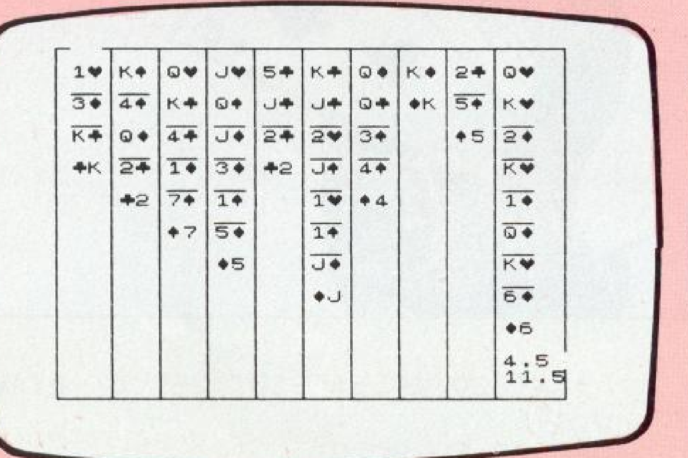

```
 .0125,y: NEXT y
3033 LET l=l+2: LET cartas=carta
s+1
3035 IF cartas>=10 THEN  LET l=l
-2
3037 LET grafico=INT (RND*4)+1
3038 LET numero=INT (RND*16)+1
3041 IF numero<8 THEN  LET p=p+n
umero
3042 IF numero>7 THEN  LET p=p+.
5
3050 IF xx=1 THEN  GO TO 3400
3060 IF l>1 THEN  GO SUB 5100
3085 PRINT  PAPER q;AT l,c;"  ";
AT l+1,c; PAPER q;"  ";AT l+2,c;
"  "
3090 IF grafico=1 THEN  PRINT  P
APER q; INK 2;AT l,c+1;"A";AT l+
2,c;"A"
3100 IF grafico=2 THEN  PRINT  P
APER q; INK 0;AT l,c+1;"B";AT l+
2,c;"B"
3110 IF grafico=3 THEN  PRINT  P
APER q; INK 2;AT l,c+1;"C";AT l+
2,c;"C"
3120 IF grafico=4 THEN  PRINT  P
APER q; INK 0;AT l,c+1;"D";AT l+
2,c;"D"
3130 IF numero=1 THEN  PRINT  PA
PER q; INK 0;AT l,c;"1";AT l+2,c
+1;"1"
3140 IF numero=2 THEN  PRINT  PA
PER q; INK 0;AT l,c;"2";AT l+2,c
+1;"2"
3150 IF numero=3 THEN  PRINT  PA
PER q; INK 0;AT l,c;"3";AT l+2,c
+1;"3"
3160 IF numero=4 THEN  PRINT  PA
PER q; INK 0;AT l,c;"4";AT l+2,c
+1;"4"
3170 IF numero=5 THEN  PRINT  PA
PER q; INK 0;AT l,c;"5";AT l+2,c
+1;"5"
3180 IF numero=6 THEN  PRINT  PA
PER q; INK 0;AT l,c;"6";AT l+2,c
+1;"6"
3190 IF numero=7 THEN  PRINT  PA
PER q; INK 0;AT l,c;"7";AT l+2,c
+1;"7"
3200 IF numero=8 THEN  PRINT  PA
PER q; INK 0;AT l,c;"J";AT l+2,c
+1;"J"
```

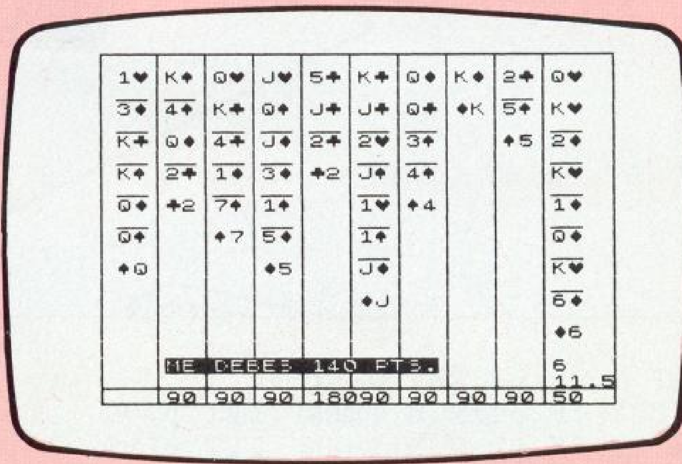
```
3210 IF numero=9 THEN  PRINT  PA
PER q; INK 0;AT l,c;"Q";AT l+2,c
+1;"Q"
3220 IF numero=10 THEN  PRINT  P
APER q; INK 0;AT l,c;"K";AT l+2,
c+1;"K"
3221 IF numero=11 THEN  PRINT  P
APER q; INK 0;AT l,c;"K";AT l+2,
c+1;"K"
3222 IF numero=12 THEN  PRINT  P
APER q; INK 0;AT l,c;"K";AT l+2,
c+1;"K"
3223 IF numero=13 THEN  PRINT  P
APER q; INK 0;AT l,c;"Q";AT l+2,
c+1;"Q"
3224 IF numero=14 THEN  PRINT  P
APER q; INK 0;AT l,c;"Q";AT l+2,
c+1;"Q"
3225 IF numero=15 THEN  PRINT  P
APER q; INK 0;AT l,c;"J";AT l+2,
c+1;"J"
3226 IF numero=16 THEN  PRINT  P
APER q; INK 0;AT l,c;"J";AT l+2,
c+1;"J"
3229 PRINT  PAPER 6; INK 0;AT 20
,28;"    ";AT 20,28;p: LET q=7
3230 PLOT 0,0: DRAW 255,0: DRAW
0,175: DRAW -255,0: DRAW 0,-175
3231 IF r=2 AND oc=1 THEN  PRINT
  PAPER 1;AT l-1,c;"  ": LET oc=
0
3232 IF cartas=1 AND numero=4 TH
EN  PRINT #1;AT 0,0;"DESEA CAMBI
AR LA CARTA POR OTRA?": PAUSE 0:
 INPUT 0
3234 IF INKEY$="S" THEN  LET r=0
: LET oc=0: GO TO 3010
3235 IF oc=1 THEN  LET r=2
3238 IF cartas=1 THEN  INPUT "CU
ANTOS DUROS JUEGAS (MAX.18)";ap
3240 IF ap>18 THEN  LET ap=0: GO
 TO 3238
3250 IF p>7.5 THEN  PRINT AT 21,
25;"PIERDE": PAUSE 200: PRINT AT
 21,25;"       ": GO TO 4000
3270 GO TO 3020
3280 STOP
3400 IF numero<8 THEN  LET cr=nu
mero
3410 IF numero>7 THEN  LET cr=.5
3420 GO TO 3060
3430 STOP
4004 FOR a=28 TO 240 STEP 24: PL
OT a,0: DRAW 0,175: NEXT a: PLOT
 0,0: DRAW 255,0: DRAW 0,175: DR
AW -255,0: DRAW 0,-175
4006 LET oc=0: LET z=z+1: LET ca
rtas=0: LET r=0
4008 IF p>7.5 THEN  LET p=0: LET
 cr=0
4009 PRINT  PAPER 4;AT 0,c;"  ":
 PLOT 0,0: DRAW 255,0: DRAW 0,17
5: DRAW -255,0: DRAW 0,-175
4010 IF z=1 THEN  LET p1=p-cr: L
ET ap1=ap: LET w1=p1+cr: PRINT
PAPER 0;AT 0,c+3;"  "
4011 IF z=2 THEN  LET p2=p-cr: L
ET ap2=ap: LET w2=p2+cr: PRINT
PAPER 1;AT 0,c+3;"  "
4012 IF z=3 THEN  LET p3=p-cr: L
ET ap3=ap: LET w3=p3+cr: PRINT
PAPER 2;AT 0,c+3;"  "
4013 IF z=4 THEN  LET p4=p-cr: L
ET ap4=ap: LET w4=p4+cr: PRINT
PAPER 3;AT 0,c+3;"  "
4014 IF z=5 THEN  LET p5=p-cr: L
ET ap5=ap: LET w5=p5+cr: PRINT
PAPER 6;AT 0,c+3;"  "
4015 IF z=6 THEN  LET p6=p-cr: L
ET ap6=ap: LET w6=p6+cr: PRINT
PAPER 7;AT 0,c+3;"  "
4016 IF z=7 THEN  LET p7=p-cr: L
ET ap7=ap: LET w7=p7+cr: PRINT
PAPER 0;AT 0,c+3;"  "
4017 IF z=8 THEN  LET p8=p-cr: L
ET ap8=ap: LET w8=p8+cr: PRINT
PAPER 1;AT 0,c+3;"  "
4018 IF z=9 THEN  LET p9=p-cr: L
```




```
ET ap9=ap: LET w9=p9+cr
4021 IF p1>7.5 THEN  LET p1=0
4022 IF p2>7.5 THEN  LET p2=0
4023 IF p3>7.5 THEN  LET p3=0
4024 IF p4>7.5 THEN  LET p4=0
4025 IF p5>7.5 THEN  LET p5=0
4026 IF p6>7.5 THEN  LET p6=0
4027 IF p7>7.5 THEN  LET p7=0
4028 IF p8>7.5 THEN  LET p8=0
4029 IF p9>7.5 THEN  LET p9=0
4030 LET xx=0: LET cr=0
4031 IF w1=7.5 THEN  LET rec=1
4032 IF w2=7.5 THEN  LET rec=2
4033 IF w3=7.5 THEN  LET rec=3
4034 IF w4=7.5 THEN  LET rec=4
4035 IF w5=7.5 THEN  LET rec=5
4036 IF w6=7.5 THEN  LET rec=6
4037 IF w7=7.5 THEN  LET rec=7
4038 IF w8=7.5 THEN  LET rec=8
4039 IF w9=7.5 THEN  LET rec=9
4040 IF z>g-1 THEN  GO TO 6000
4045 IF z=g THEN  PRINT  PAPER 4
;AT 0,c+3;"   "
4047 PRINT #1;AT 0,0;"PULSE UNA
TECLA PARA CONTINUAR": PAUSE 0:
INPUT 0
4050 LET c=c+3: LET p=0: LET l=-
1: LET ap=0
4055 PRINT  PAPER 6;AT 20,28;"
  ";AT 20,28;p: PLOT 255,0: DRAW
 0,175
4060 GO TO 3020
4200 PRINT #1;AT 0,0;"***VISTA (
V) O SE PLANTA (P)*** ": PAUSE 0
: INPUT 0
4210 GO TO 3027
5100 PRINT  PAPER 7;AT l-1,c;"__
"
5110 RETURN
6000 REM Juego del ordenador
6001 LET t=0: LET s=0: LET f=0:
LET f1=0: LET f2=0: LET f3=0: LE
T f4=0: LET f5=0: LET f6=0: LET
f7=0: LET f8=0: LET f9=0
6002 LET s1=0: LET s2=0: LET s3=
0: LET s4=0: LET s5=0: LET s6=0:
 LET s7=0: LET s8=0: LET s9=0
6003 PRINT  PAPER 4;AT 0,c+3;"
": PLOT 0,0: DRAW 255,0: DRAW 0,
175: DRAW -255,0: DRAW 0,-175
6005 PRINT  PAPER 2;AT 0,1;"  "
6010 LET cartas=0: LET l=-1: LET
 c=1: LET po=0
6015 PAUSE 100
6016 FOR x=-10 TO 0: BEEP .0125,
x: NEXT x
6017 FOR y=0 TO -5 STEP -1: BEEP
 .0125,y: NEXT y
```

```
6030 LET grafico=INT (RND*4)+1
6040 LET numero=INT (RND*16)+1
6045 LET l=l+2
6050 IF numero<8 THEN  LET po=po
+numero
6060 IF numero>7 THEN  LET po=po
+.5
6065 LET cartas=cartas+1
6070 IF cartas>=10 THEN  LET l=l
-2
```






```
6080 IF l>2 THEN  GO SUB 7000
6110 PRINT  PAPER 7;AT l,c;"   ";
AT l+1,c; PAPER 7;"   ";AT l+2,c;
"   "
6120 IF grafico=1 THEN  PRINT  P
APER 7; INK 2;AT l,c+1;"A";AT l+
2,c;"A"
6130 IF grafico=2 THEN  PRINT  P
APER 7; INK 0;AT l,c+1;"B";AT l+
2,c;"B"
6140 IF grafico=3 THEN  PRINT  P
APER 7; INK 2;AT l,c+1;"C";AT l+
2,c;"C"
6150 IF grafico=4 THEN  PRINT  P
APER 7; INK 0;AT l,c+1;"D";AT l+
2,c;"D"
6160 IF numero=1 THEN  PRINT  PA
PER 7; INK 0;AT l,c;"1";AT l+2,c
+1;"1"
6170 IF numero=2 THEN  PRINT  PA
PER 7; INK 0;AT l,c;"2";AT l+2,c
+1;"2"
6180 IF numero=3 THEN  PRINT  PA
PER 7; INK 0;AT l,c;"3";AT l+2,c
+1;"3"
6190 IF numero=4 THEN  PRINT  PA
PER 7; INK 0;AT l,c;"4";AT l+2,c
+1;"4"
6200 IF numero=5 THEN  PRINT  PA
PER 7; INK 0;AT l,c;"5";AT l+2,c
+1;"5"
6210 IF numero=6 THEN  PRINT  PA
PER 7; INK 0;AT l,c;"6";AT l+2,c
+1;"6"
6220 IF numero=7 THEN  PRINT  PA
PER 7; INK 0;AT l,c;"7";AT l+2,c
+1;"7"
6230 IF numero=8 THEN  PRINT  PA
PER 7; INK 0;AT l,c;"J";AT l+2,c
+1;"J"
6240 IF numero=9 THEN  PRINT  PA
PER 7; INK 0;AT l,c;"Q";AT l+2,c
+1;"Q"
6250 IF numero=10 THEN  PRINT  P
APER 7; INK 0;AT l,c;"K";AT l+2,
c+1;"K"
6251 IF numero=11 THEN  PRINT  P
APER 7; INK 0;AT l,c;"K";AT l+2,
c+1;"K"
6252 IF numero=12 THEN  PRINT  P
APER 7; INK 0;AT l,c;"K";AT l+2,
c+1;"K"
6253 IF numero=13 THEN  PRINT  P
```

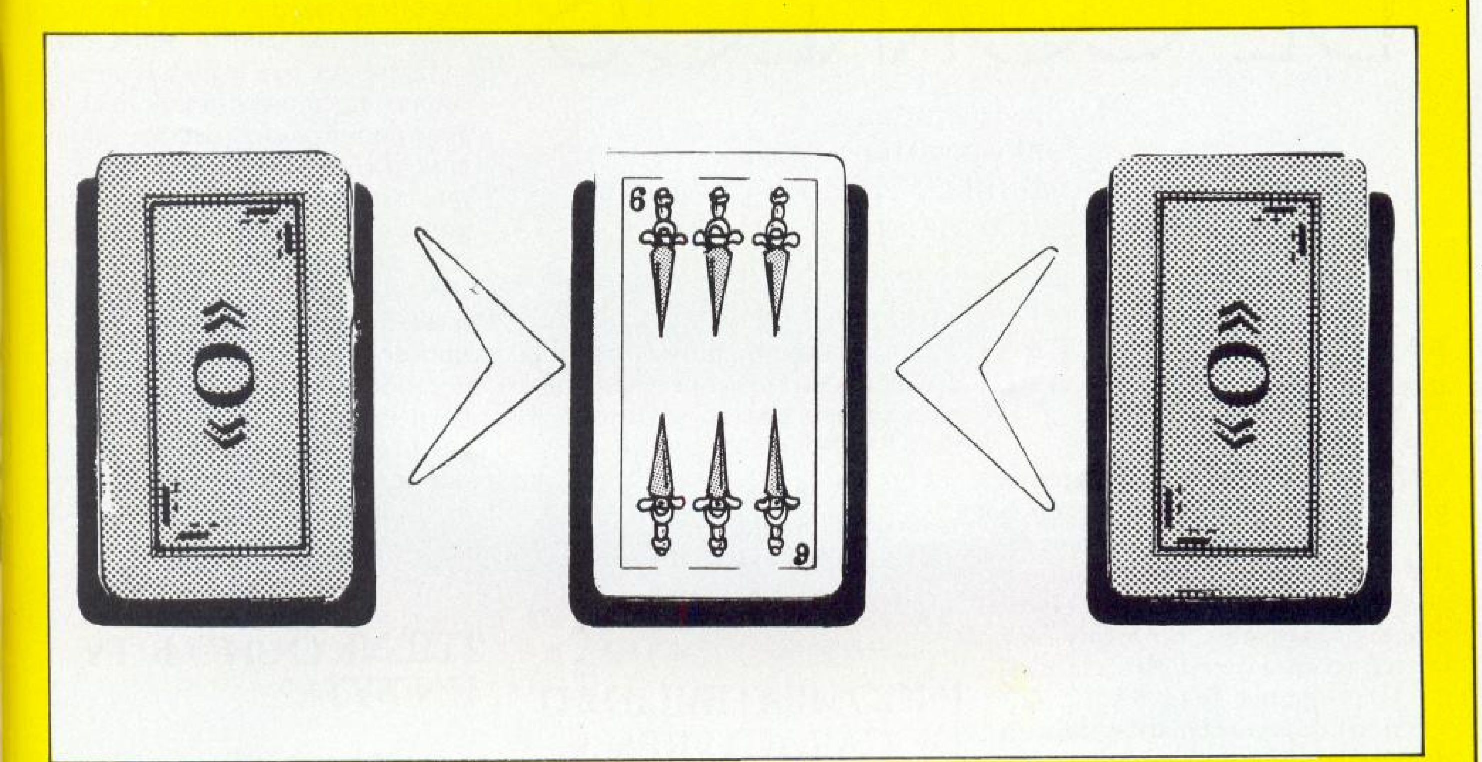
```
APER 7; INK 0;AT l,c;"Q";AT l+2,
c+1;"Q"
6254 IF numero=14 THEN  PRINT  P
APER 7; INK 0;AT l,c;"Q";AT l+2,
c+1;"Q"
6255 IF numero=15 THEN  PRINT  P
APER 7; INK 0;AT l,c;"J";AT l+2,
c+1;"J"
6256 IF numero=16 THEN  PRINT  P
APER 7; INK 0;AT l,c;"J";AT l+2,
c+1;"J"
6257 PRINT  PAPER 2; INK 0;AT 19
,28;"    ";AT 19,28;po
6258 IF po>7.5 THEN  LET po=0: G
O TO 6301
6259 IF po=7.5 THEN  GO TO 6301
6261 IF po<=p1 THEN  GO TO 6015
6262 IF po<=p2 THEN  GO TO 6015
6263 IF po<=p3 THEN  GO TO 6015
6264 IF po<=p4 THEN  GO TO 6015
6265 IF po<=p5 THEN  GO TO 6015
6266 IF po<=p6 THEN  GO TO 6015
6267 IF po<=p7 THEN  GO TO 6015
6268 IF po<=p8 THEN  GO TO 6015
6269 IF po<=p9 THEN  GO TO 6015
6301 IF po>=w1 THEN  LET s1=5*ap
1: PRINT  PAPER 2;AT 21,4;s1
6302 IF po>=w2 THEN  LET s2=5*ap
```

```
2: PRINT  PAPER 2;AT 21,7;s2
6303 IF po>=w3 THEN  LET s3=5*ap
3: PRINT  PAPER 2;AT 21,10;s3
6304 IF po>=w4 THEN  LET s4=5*ap
4: PRINT  PAPER 2;AT 21,13;s4
6305 IF po>=w5 THEN  LET s5=5*ap
5: PRINT  PAPER 2;AT 21,16;s5
6306 IF po>=w6 THEN  LET s6=5*ap
6: PRINT  PAPER 2;AT 21,19;s6
6307 IF po>=w7 THEN  LET s7=5*ap
7: PRINT  PAPER 2;AT 21,22;s7
6308 IF po>=w8 THEN  LET s8=5*ap
8: PRINT  PAPER 2;AT 21,25;s8
6309 IF po>=w9 THEN  LET s9=5*ap
9: PRINT  PAPER 2;AT 21,28;s9
6311 IF po<w1 THEN  LET f1=5*ap1
: PRINT  PAPER 6;AT 21,4;f1
6312 IF po<w2 THEN  LET f2=5*ap2
: PRINT  PAPER 6;AT 21,7;f2
6313 IF po<w3 THEN  LET f3=5*ap3
: PRINT  PAPER 6;AT 21,10;f3
6314 IF po<w4 THEN  LET f4=5*ap4
: PRINT  PAPER 6;AT 21,13;f4
6315 IF po<w5 THEN  LET f5=5*ap5
: PRINT  PAPER 6;AT 21,16;f5
6316 IF po<w6 THEN  LET f6=5*ap6
: PRINT  PAPER 6;AT 21,19;f6
6317 IF po<w7 THEN  LET f7=5*ap7
: PRINT  PAPER 6;AT 21,22;f7
6318 IF po<w8 THEN  LET f8=5*ap8
: PRINT  PAPER 6;AT 21,25;f8
6319 IF po<w9 THEN  LET f9=5*ap9
: PRINT  PAPER 6;AT 21,28;f9
6321 IF po<w1 AND w1=7.5 THEN  L
ET f1=10*ap1: PRINT  PAPER 6;AT
21,4;f1
6322 IF po<w2 AND w2=7.5 THEN  L
ET f2=10*ap2: PRINT  PAPER 6;AT
21,7;f2
6323 IF po<w3 AND w3=7.5 THEN  L
ET f3=10*ap3: PRINT  PAPER 6;AT
21,10;f3
6324 IF po<w4 AND w4=7.5 THEN  L
ET f4=10*ap4: PRINT  PAPER 6;AT
21,13;f4
6325 IF po<w5 AND w5=7.5 THEN  L
ET f5=10*ap5: PRINT  PAPER 6;AT
21,16;f5
6326 IF po<w6 AND w6=7.5 THEN  L
ET f6=10*ap6: PRINT  PAPER 6;AT
21,19;f6
6327 IF po<w7 AND w7=7.5 THEN  L
ET f7=10*ap7: PRINT  PAPER 6;AT
21,22;f7
6328 IF po<w8 AND w8=7.5 THEN  L
ET f8=10*ap8: PRINT  PAPER 6;AT
21,25;f8
6329 IF po<w9 AND w9=7.5 THEN  L
ET f9=10*ap9: PRINT  PAPER 6;AT
21,28;f9
6330 LET s=s1+s2+s3+s4+s5+s6+s7+
s8+s9
6333 LET f=f1+f2+f3+f4+f5+f6+f7+
f8+f9
6335 LET t=f-s
6340 IF s>f THEN  PRINT  PAPER 3
;AT 19,4;"ME DEBES ";-t;" PTS.";
n$
6350 IF s<f THEN  PRINT  PAPER 3
;AT 19,4;"TE DEBO ";t;" PTS.";n$

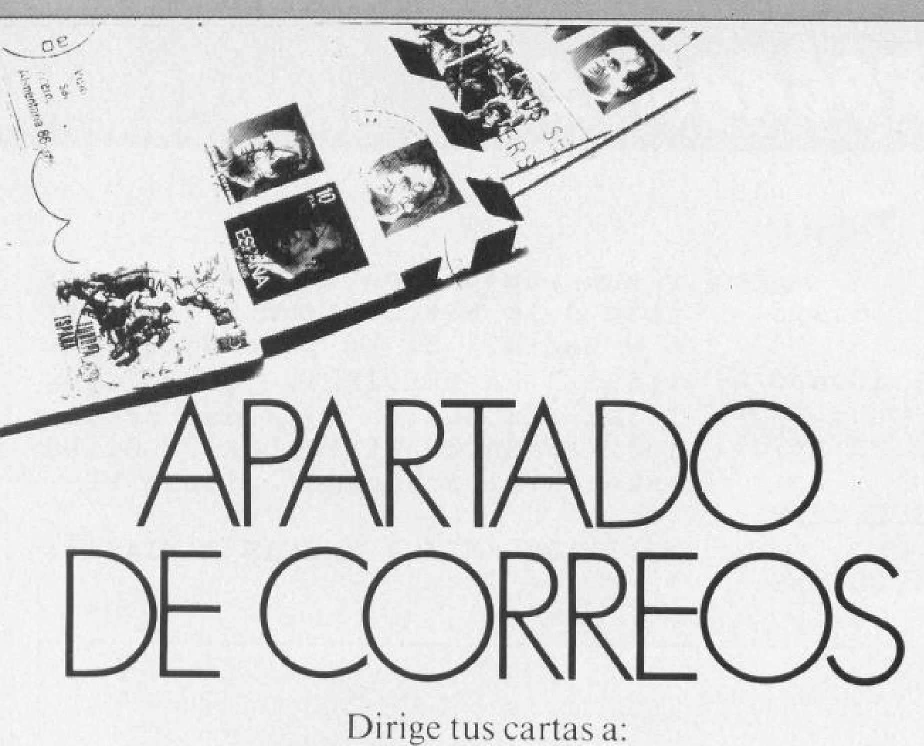
6360 IF s=f THEN  PRINT  PAPER 3
;AT 19,4;"ESTAMOS EN PAZ.";n$
6370 PLOT 0,0: DRAW 255,0: DRAW
0,175: DRAW -255,0: DRAW 0,-175
6375 PLOT 0,8: DRAW 255,0
6380 PRINT #1;AT 0,0; INK 2;"PUL
SE UNA TECLA PARA CONTINUAR": PA
USE 0: INPUT 0
6390 CLS : GO TO 1500
6399 STOP
6400 PRINT  PAPER 6;AT 21,4;ap1*
5
6401 PRINT  PAPER 6;AT 21,7;ap2*
5
6402 PRINT  PAPER 6;AT 21,10;ap3
*5
6403 PRINT  PAPER 6;AT 21,13;ap4
```

```
*5
6404 PRINT  PAPER 6;AT 21,16;ap5
*5
6405 PRINT  PAPER 6;AT 21,19;ap6
*5
6406 PRINT  PAPER 6;AT 21,22;ap7
*5
6407 PRINT  PAPER 6;AT 21,25;ap8
*5
6408 PRINT  PAPER 6;AT 21,28;ap9
r una puntuacion lo mas alta pos
ible o lo mas cercana a las siet
e y media.  Si Vd pasa dicha can
tidad, !HA PERDIDO!.  Si Vd igua
la su puntuacion a la del ordena
dor, !GANA EL ORDENADOR!.  Si Vd
 supera al ordenador, !GANA VD!.

  !!!ADELANTE Y BUENAS CARTAS!!!
"
```



```
*5
6500 LET s=(ap1+ap2+ap3+ap4+ap5+
ap6+ap7+ap8+ap9)*5: PRINT AT 20,
7;"TE DEBO ";s;" PTS.": PLOT 0,0
: DRAW 255,0: DRAW 0,175: DRAW -
255,0: DRAW 0,-175: PAUSE 0: GO
TO 1500
7000 PRINT  PAPER 7;AT l-1,c;"__
"
7010 RETURN
8860 REM Presentacion Programa
8865 PAPER 0: CLS
8870 LET a$="   Este programa tr
ata del conocido juego de cartas
..................******LAS SIETE
 Y MEDIA******
                Vd, debera ganar a
l ordenador tratando de consegui
8881 LET x=10: LET y=1: LET z=2
8882 BORDER 2: CLS : DIM b$(25)
8883 LET a$=b$+a$
8884 FOR n=1 TO LEN a$
8885 LET a$=a$(2 TO )+CHR$ 32
8886 PRINT AT x,y; INK 7; PAPER
3;a$(z TO 31)
8887 BEEP .01,20
8889 NEXT n
8890 BORDER 4: PAPER 7: INK 0: C
LS
8900 PRINT  INK 2;AT 10,0;"   De
sea volver a leer las ins- trucc
iones S/N ?"
8920 PAUSE 0
8940 IF INKEY$="S" OR a$="s" THE
N  GO TO 8870
8950 RETURN
```



APARTADO DE CORREOS

Dirige tus cartas a:
Todospectrum
Bravo Murillo, 377, 5.º-A
28020 Madrid

EN BUSCA DEL BETA BASIC PERDIDO

Quisiera, por un lado, falicitaros por la labor que desarrolláis en la revista, que me parece de una calidad y nivel técnico muy elevados, y por otra parte, pediros un favor: que, si os es posible, me localicéis y enviéis (contra reembolso, claro está) el programa Beta BASIC (2.ª Versión) que comentabais en uno de vuestros últimos números, ya que no me ha sido posible conseguirlo en los sitios donde lo he buscado, que no han sido pocos (Puertollano, Ciudad Real, Granada).

También querría que me informaseis si conocéis algún compilador Fortran para el Spectrum 48K, semejante al de Pascal de Hisoft, que ya poseo. En el caso de que exista y podáis conseguirlo, ¿podríais mandármelo igualmente contrareembolso? En caso de que vosotros no podáis localizármelo y/o mandármelo, dadme una dirección de algún sitio donde lo puedan tener, pues tampoco lo he visto en ninguno de los anuncios comerciales de la revista.

Pablo Higueras
Almadén (C. Real)

No conocemos ningún compilador de Fortran para Spectrum que se haya comercializado en España. El Beta BASIC puedes pedirlo a:

Ventamatic
Córcega, 89
08029 Barcelona

INCOMPATIBILIDAD DE CARACTERES

Tengo una impresora New Print, el interface serie/paralelo de Indescomp y el Tasword Two, y no consigo hacer funcionar la impresora con el Tasword; alguna vez me ha sacado algunos caracteres que no se correspondían con los escritos, y con el manual no hago nada porque está en inglés y no entiendo nada. Donde lo compré no saben, no contestan, y me encuentro un tanto colgado. Esta carta está hecha con la impresora pero usando el editor que publicásteis vosotros, aunque me resultra un lio cuadrar el texto en pantalla a como me lo saca la impresora.

Antonio Diaz
Lérida

A la hora de comprar algo complejo de manejar y que vale una cierta «pasta», como es una impresora, es evidente que no se puede andar uno con tonterías. A no ser que se sepa con seguridad lo que se compra y la forma en que vamos a utilizarlo, es bastante aconsejable siempre el acudir a una tienda especializada en «cacharros» de este tipo, donde nos puedan asesorar sobre cuáles entre las diversas marcas y modelos que ofrece el mercado se acerca más a lo que necesitamos, así cómo resolvernos los problemas que nos surjan a posteriori (el llamado servicio postventa).

Parece ser que o no adquiriste la impresora en un sitio serio o el sitio serio donde la adquiriste es cualquier cosa menos serio. Te aconsejamos que insistas en que te busquen allí algún tipo de arreglo que te permita usar ese u otro procesador de textos con tu impresora. Nuestras páginas quedan siempre abiertas a cualquier tipo de acusación pública si siguen negándose a hacer nada por tí, en cuyo caso tendrás que buscar a alguien que sepa algo de inglés para que te ayude con el manual.

¿TIENE OCHO BITS UN BYTE?

Quisiera hacerles una pregunta elemental que probablemente ya les habrán hecho: ¿qué es exactamente un byte?; me explico, cuando un ordenador con un micro de 16 bits dicen que tiene, por ejemplo, un Kbyte, ¿tiene 1024 palabras de 16 o son éstas de ocho bits? No estoy muy seguro, pero creo haber leido informaciones contradictorias al respecto.

Juan M. Villar
Mardid

También nosotros nos hemos encontrado con algunos escritos contradictorios cuando se toca este tema, aunque son debidos más a traducciones del inglés «poco inteligen-

tes» que a otra cosa. Hoy por hoy debemos hablar de byte como un conjunto de ocho bits, eso que debería haber sido traducido en su día como octeto (hubo quien lo hizo) y que hoy forma parte del vocabulario de cualquier aficionado a la informática. Se suele hablar de «palabra» en un ordenador como el conjunto de bits que es capaz de manejar su microprocesador dependiendo del lbus de datos que utilice. Así las «palabras» en el Spectrum serían de un byte, mientras que en un ordenador que utilice un 16 bits serían de dos bytes.

GENS Y MONS RELOCALIZABLES

¿Cómo puedo hacer para que el GENS y el MONS sean relocalizables?, ¿cómo podría hacer para que un programa mio sea relocalizable si tiene mnemónicos como son JP y CALL?

César Blanco
Barcelona

Que nosotros sepamos, el GENS y el MONS forman uno de los pocos paquetes ensamblador-monitor totalmente relocalizables, efectuándose esta relocalización cuando los ejecutamos en la dirección de comienzo (para reentrar tras haber vuelto al BASIC hay que saltar a esa dirección +61 para el GENS y +29 para el MONS).

Puedes relocalizar tus programas sin ningún problema si conservas el código fuente y utilizas etiquetas en todas las llamadas y saltos absolutos que incluya; de esta forma bastará con ensamblar en la nueva dirección. Si no usamos etiquetas o no disponemos del código fuente la cosa se complica, pues habrá que hacerlo «a brazo», a no ser que dispongamos de un programa (es factible aunque no conocemos ninguno para Spectrum) que se encargue de seguir todo el flujo del programa modificando (sólo en la primera pasada) los CALLs y JumPs con que se encuentre.

UN AMIGO DE QL

En primer lugar quiero agradeceros la labor que hacéis en cuanto a la difusión y mejor conocimiento del ordenador olvidado de Sinclair, el QL, ya que vuestra revista es el único vínculo de unión entre la gran mayoría de los usuarios del QL,. que andan sin conocer a nadie y sin nadie que les cambie programas, ideas, etc.

Hace algún tiempo, en la revista ZX se publicó un anuncio de un usuario de QL, pero el escaso eco entre los que tienen un QL fue más que evidente: sólo 22 personas de toda Espana (entre ellas yo) contestaron a su llamada. Por ello me dirijo a vuestra revista para ver si así conseguimos contactar con más usuarios del QL. Mi dirección es:

Manuel José Garrido
San Vicente de Paul, 25, pta. 36
46019 Valencia
Tel. (97) 365 63 96

Sin otro particular, y esperando que continuéis publicando el suplemento QL en vuetra revista, se despide un lector vuestro.

Manuel J. Garrido
Valencia

Aunque quizá deberías haber dirigido tu carta a nuestra sección de anuncios gratuitos, «el Corcho», nos alegra ver que hay QLmaníacos como tú con ganas de que a su ordenador no le salgan telarañas. Nos alegraríamos muchos más aún si fueran miles las cartas que te llegaran de otros usuarios de QL dispuestos a «mover» su máquina.

AMPLIACION DE MEMORIA

Tengo un Spectrum 16K y quisiera ampliarlo a 48K. Me han dicho que existe una ampliación de memoria externa que es muy sencilla de conectar y que no anula la garantía del ordenador. ¿Esto es así?, ¿cuál es su precio?

Raul Martínez
Gijón (Asturias)

La ampliación de memoria externa es, efectivamente, muy sencilla de conectar, pues no es necesario desmontar el ordenador para hacerlo, sino que basta con «pincharla» al bus de expansión como un periférico más. Como contrapunto está el que pueda molestar el llevarla «colgando» todo el tiempo, además de que la conexión sufre un tanto al estar expuesta a los movimientos del bloque. Su precio está alrededor de las 8.000 pesetas.

SOBRE LOS REGISTROS IX E IY

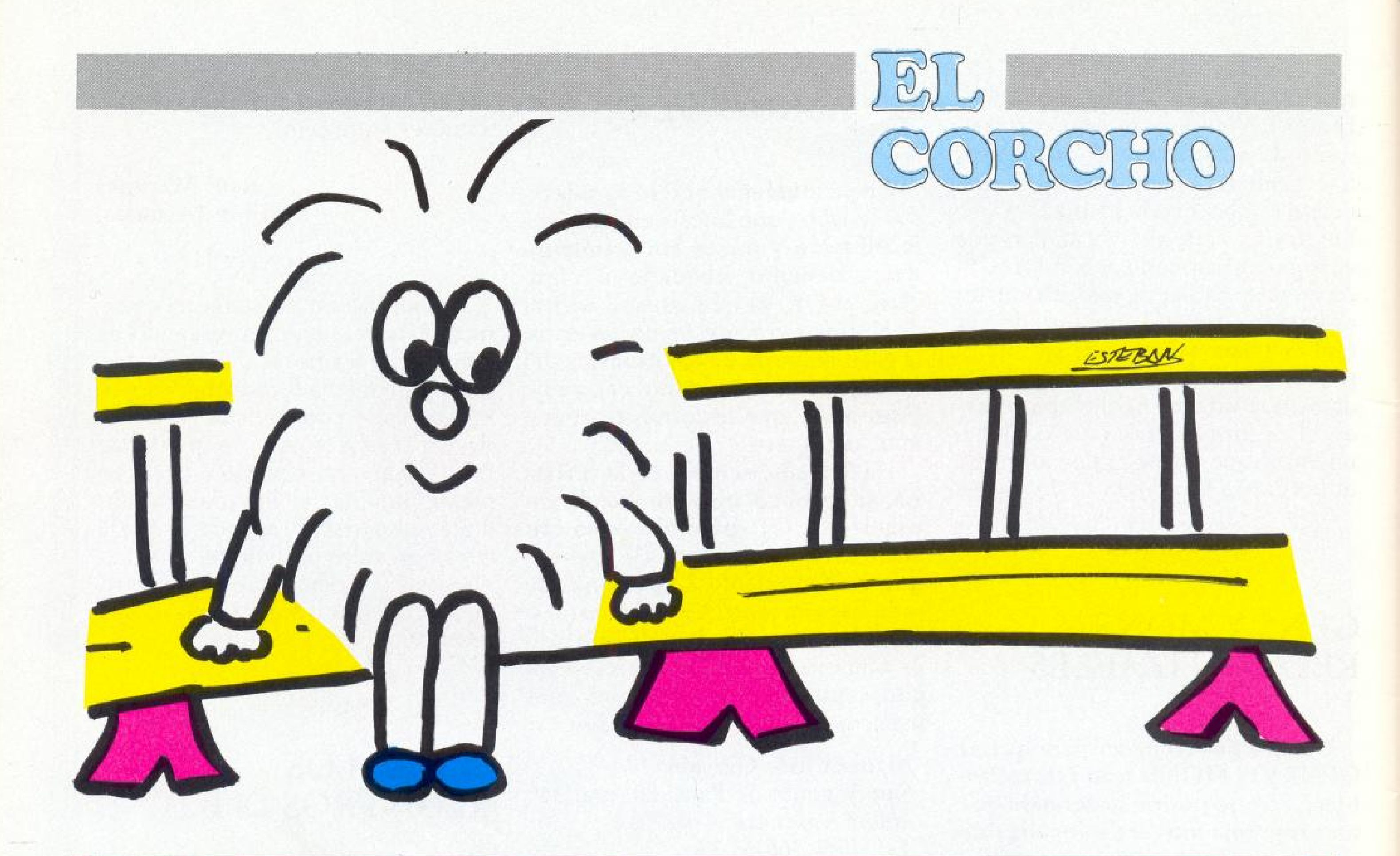
¿Podemos (en código máquina) manejar directamente los registros IX e IY? Y si no, ¿para qué sirven? ¿Cómo se usan las instrucciones BIT, CCF, DI, IND, LDD, NEG, NOP, RES y SET?

Javier Burgués
Lérida

Puedes utilizar los registros IX e IY con toda comodidad en tus programas en ensamblador, admitirán todas las instrucciones que sueles utilizar con el par HL, pero cuando direccionen alguna posición de memoria, deberán ir seguidos de un byte en complemento a dos. Por ejemplo, puedes usar perfectamente LD A,(IX+3). Has de tener más cuidado con el par IY, pues es utilizado por el operativo para indexar las variables del sistema.

En cuanto a la «ristra» de instrucciones sobre las que pides consejo, es evidente que no es ésta la sección adecuada para tan extenso tema. Te recomendamos que sigas la serie «Aprendiendo lenguaje máquina» en la que si no han sido ya vistas esas instrucciones lo serán pronto.

EL CORCHO



Vendo programas de QL, comerciales, juegos y utilidades, al precio de 1.100-1.500 ptas., incluyendo microdrive. Interface conector de QL a impresora paralelo. Interface conector Spectrum a paralelo. Cable extensor paralelo. Carro de tracción para impresora Smith-Corona y bandeja de papel para la misma. Jacques Bulchand. Avda. Primero de Mayo, 6. 35002 Las Palmas. Tel. (928) 369862.

Vendo calculadora de bolsillo Casio PB-100 programable en BASIC, con instrucciones en castellano, ampliación de memoria, interface para cassette y varios programas de utilidades. Todo 10.000 ptas. Francisco Martínez. Vilamarí, 33, pral. 1.ª. 08015 Barcelona. Tel. (93) 2241113.

Vendo impresora Seikosha GP-500 a muy buen precio. Daniel Sáez. Tel. (93) 2460761.

Vendo impresora GP-50S con muy poco uso y prácticamente nueva, con todos los accesorios y embalaje original, con un rollo de papel y varias cintas de juegos. Todo por sólo 18.000 ptas. (negociables). Escribir a José Antonio Rodríguez Ovalle. Apdo. Correos 28. 04080 Villafranca del Bierzo (León).

Deseo contactar con usuarios del QL para todo tipo de intercambio de programas, publicaciones, etc. Dispongo de abundante software y hardware. Albert Busoms. Tel. (93) 2130153.

QL: todos los programas. Pascal, Fortran-77, compilador SuperBASIC, lenguaje C, Nóminas, utilidades, juegos, copiadores, importados,... Vendo, compro o cambio. Juan Carlos Ordóñez. Ferroviarios, 11, 3.º C. 28026 Madrid. Tel. (91) 4762539.

¿Quieres un Spectrum 48K, cinta de demostración Horizontes, manual en castellano, joystick con interface Kempston, originales de Astrodata 3000, Hypersports, Divertimentos, 3d monstruos y varias cintas con más de 100 programas comerciales? Pues sólo te costará 20.000 ptas. Asier Burgaleta. Añorga Txiki, 9, 2.º C. 20009 Donostia.